A mis padres. Y, por supuesto, a mi niña.

Abstract

The purpose of this thesis is to present a comprehensive and practical approach for the time-optimal motion planning and control of a general purpose industrial manipulator. In particular, the case of point-to-point path unconstrained motions is considered, with special emphasis towards strategies suitable for efficient on-line implementations. From a dynamic model description of the plant, and using an advanced graphical robotics simulation environment, the control algorithms are formulated. Experimental work is then conducted to verify the proposed algorithms, by interfacing the industrial manipulator to the master controller, implemented on a personal computer.

The full rigid-body non-linear dynamics of the open-chain manipulator have been accommodated into the modelling, analysis and design of the control algorithms. For path unconstrained motions, this leads to a model-based regulating strategy between set points, which combines conventional trajectory planning and subsequent control tracking stages into one. Theoretical insights into these two robot motion disciplines are presented, and some are experimentally demonstrated on a CRS A251 industrial arm.

A critical evaluation of current approaches which yield optimal trajectory planning and control of robot manipulators is undertaken, leading to the design of a control solution which is shown to be a combination of Pontryagin's Maximum Principle and state-space methods of design. However, in a real world setting, consideration of the relationship between optimal control and on-line viability highlights the need to approximate manipulator dynamics by a piecewise linear and decoupled function, hence rendering a near-time-optimal solution in feedback form.

The on-line implementation of the proposed controller is presented together with a comparison between simulation and experimental results. Furthermore, these are compared with measurements from the industrial controller. It is shown that the model-based near-optimal-time feedback control algorithms allow faster manipulator motions, with an average speed-up of 14%, clearly outperforming current industrial controller practices in terms of increased productivity. This result was obtained by setting an acceptable absolute error limit on the target location of the joint (position and velocity) to within [2.0E-02 rad, 8.7E-03 rad/s], when the joint was regarded at rest.

Contents

Acknowledgments

I would like to thank my supervisors, Dr. A.S. White and Dr. Mark Stoker, for their advice and guidance during the course of this research. Other members of the staff who have willingly and consistently helped me are Dr. J.B. Lewis, Dr. M. Karamanoglu, Mr. P.R. Warner, Dr. J.S. Lewis, and my Director of Studies, Dr. R. Gill.

I wish to express my sincerest gratitude for their constant encouragement and assistance to my fellow research students at the Advanced Manufacturing and Mechatronics Centre (AMMC) at Middlesex University – Mr. J.S. "Pali boy" Surdhar, Mr. B. "Berni" Parsons, Mr. J. "John the Finn" Korhonen (no longer at the AMMC), Dr. S.D. "Dr. Flyer" Prior (no longer at the AMMC), Mr. R.R. Chanmugam, Mr. S. Okrongli (no longer at the AMMC) and Mr. A. Lasebe. In particular my good friend Pali, with whom I have shared intensive research discussions about model-based and model-free control strategies, sushi and pub rounds, and Berni, who patiently helped to fill my "electronic black-holes" during the development of the interface. Worthy of special mention is Dr. J.B. Lewis, AMMC Research Fellow and fellow colleague indeed, for reading preliminary versions of this manuscript and helping to shape it in its current form, and also for his excellent assistance in all matters concerning the robot manipulator. I am also indebted to Dr. J.S. Lewis for kindly agreeing to proofread draft copies of this dissertation and providing so many constructive comments.

I would like to thank the following bodies formally for financial aid I received from them during my stay at Middlesex University: The AMMC, The Higher Education Funding Council for England (HEFCE) for a research studentship and British Nuclear Fuels Plc. UK, for funding the purchase of the graphical simulation software and hardware utilised in this project. In particular, Mr. G. Wilson and J.A. Schofield for their continuous interest in both, my PhD research work and my research training. I am also grateful for travel grants received from BNF Plc. UK, The Royal Society (London), The British Council (Tokyo), The European Union (EU) under the Human Capital and Mobility (HCM) programme and The London and South East High Performance Computing (SEL-HPC).

I should also thank the Mechatronics Lab technician – Mr. I. Bahiji – and the Electronic technicians – Mr. I. Siman and Mr. K. Bone – for their help during the course of this research. I also wish to thank Mr. R. Gledhill for all issues related to LaTeX, and Mr. T. Schlichter for his HP UNIX support.

The critical feedback provided by anonymous referees who reviewed the papers submitted to international conferences has also been of great encouragement to me in confirming the research direction being taken, even if opinions about the work carried out were not always unanimous!. I would also like to show my gratitude to Prof. Yurkovich for opening my eyes to the current status of stability issues in the non-linear control community. His valuable point of view about "technology leading the theory" in model-free non-linear analysis proved an interesting and reassuring thought, which has ultimately reflected back into my own work.

Very many thanks to Professor J. Tornero i Montserrat and Dr. J. Picó Marco, my robotics and control lecturers at my hometown university in Valencia (Spain), for their continuous support of my career along this path I have chosen, and their willingness in setting up collaborative links between both universities. A warm thanks also to S. "Gary" Kobayashi, M. "Moto" Hattori, R. Kanno, Dr. S. Tadokoro, Prof. T. Takamori and all the graduate students at the Dept. of Computer and Systems Engineering at Kobe University, for hosting us for a wonderful and constructive month during a collaborative research project with Kobe University, Japan. Special thanks must also go to my old flat-mate and research colleague, Mr. M. "Mickey" Mullins, with whom I have shared endless hours of conversation about university- and non-university-related issues, and for introducing me to the joy of a *real* Guinness in his homeland. Also, special thanks to my friend A. Wheeler just for being there and listening on the many occasions that things went wrong.

And it is certainly with the greatest pleasure that I thank my family and close friends back at home, who have always been, sometimes not without pain, loving and supporting in my academic endeavours in this far-away land that has warmly fostered me for the last years.

This thesis is based on work undertaken between April 1994 and October 1997. The work has been carried out in the School of Engineering Systems at the Faculty of Technology.

List of Figures

List of Tables

Notation

$\boldsymbol{A}$	State or system matrix
$\boldsymbol{B}$	Input matrix
B	Viscous friction coefficient
$\boldsymbol{C}$	Output matrix
c_i	Sliding gain
$\boldsymbol{D}$	Direct transmission matrix
$\boldsymbol{D(q)}$	Inertia matrix
$\boldsymbol{D_{eff}}$	Effective inertia matrix
E_{ss}	Manipulator steady-state error
$\boldsymbol{e}$	Position or tracking error matrix
e	Position or tracking error scalar
$\boldsymbol{F}$	Kalman filter feedback matrix
F_s	Starting static friction or stiction
$\boldsymbol{f}$	State matrix function
f	Friction
$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}$	Partial derivative of f with respect to $\boldsymbol{x}$
$\boldsymbol{G(q)}$	Gravity matrix
G	Performance index integrand or loss function
$\boldsymbol{g}$	Output matrix funtion
$\boldsymbol{H(q,\dot{q})}$	Centripetal and Coriolis matrix
$\mathcal{H}$	Hamiltonian
I_f	Field current
I_m	Armature current
$\boldsymbol{J}$	Jacobian matrix
J	Performance index
$J_{k,k+1}$	Performance measure between stage k and $k+1$
$J^*_{k,N}$	Best performance measure from stage k to final stage N
J_m	Mass moment of inertia of motor
$\boldsymbol{K}$	Gain matrix
$\boldsymbol{K_p}$	Proportional gain matrix
$\boldsymbol{K_d}$	Derivative gain matrix
K	Kinetic energy
K_{amp}	Amplifier gain
K_{bemf}	Back e.m.f. motor constant
K_d	Damping constant
K_f	Flux constant
K_i	Integral constant
K_m	Motor-torque constant
K_p	Pure gain constant
L	Lagrangian function
L_f	Field inductance
L_m	Armature inductance
M_p	Maximum % overshoot

N	Gear ratio
N_l	Number of teeth on the link gear
N_m	Number of teeth on the motor gear
$O(n)$	Computational complexity of order n
$\boldsymbol{p}$	Lagrange multiplier matrix
$\boldsymbol{p}^T$	Transpose of matrix $\boldsymbol{p}$
P	Potential energy
p_i	Lagrange multiplier component
$\boldsymbol{q}$	Generalised coordinate matrix
q_i	Joint i generalised coordinate
$\boldsymbol{R}$	Tool orientation or rotation matrix
R_f	Field resistance
R_m	Armature resistance (including brushes)
$\boldsymbol{r}$	Tool position matrix
$\{\boldsymbol{r},\boldsymbol{R}\}$	Tool configuration pair
$\Re$	The real line
$\boldsymbol{s}$	Slide mode control trajectory matrix
s_i	Joint i slide mode control trajectory
s	Laplace complex variable
S	Distance parameter along a path
$\boldsymbol{T}$	Homogeneous transformation matrix
T_m	Motor time constant
t_{ss}	Time to reach the steady-state position
t_{total}	Maximum of t_{ss} for all the joints
$\boldsymbol{U}$	Control closed bounded admissible region
U_{max}	Absolute upper bound of control admissible region
U_{min}	Absolute lower bound of control admissible region
$+U$	Instantaneous upper bound of control admissible region
$-U$	Instantaneous lower bound of control admissible region
$\boldsymbol{u}$	Input matrix
$\boldsymbol{u}^*$	Optimal value of $\boldsymbol{u}$
u_i	Input matrix component
u_{util}	Torque utilisation
$\|u\|$	Absolute value of u
V_{bemf}	Back e.m.f. voltage
V_m	Armature voltage
$\boldsymbol{v}$	Time derivative of $\boldsymbol{e}$
$\boldsymbol{w}$	Tool-configuration matrix
$\boldsymbol{x}$	State matrix
x_i	State matrix component
$\dot{\boldsymbol{x}}$	Time derivative of $\boldsymbol{x}$
$\ddot{\boldsymbol{x}}$	Second time derivative of $\boldsymbol{x}$
$\{\alpha_i,\beta_i\}$	Joint i piecewise-linear dynamic coefficients
$\{\overline{\alpha}_i,\overline{\beta}_i\}$	Averaged joint i piecewise-linear dynamic coefficients
$\delta\boldsymbol{e}$	Differential change in $\boldsymbol{e}$
η	Gear efficiency
$\boldsymbol{\theta}$	Joint angle matrix
θ_i	Joint i angle
λ	Averaged dynamics weighting coefficient
$\boldsymbol{\tau}$	Control torque matrix
τ	Torque dissipated at the link shaft
τ_i	Control torque component
τ_m	Torque dissipated at the motor shaft

τ_{total}	Torque developed at the motor
τ^*	Motor load torque referred to the motor shaft
ψ	Air gap flux
Δt	Sampling time interval
$\mathbf{\Lambda}$	Direct kinematic transformation
$\mathbf{\Lambda}^{-1}$	Inverse kinematic transformation
$\mathbf{\Omega}$	Real plant dynamics
$\hat{\mathbf{\Omega}}$	Model of plant dynamics
$\hat{\mathbf{\Omega}}^{-1}$	Inverse dynamic mapping
∞	Infinity

Glossary of Terms

AC	Alternating current
ACSL	Advanced Continuous Simulation Language
ACW	Adjustable control-variation weight
ADC	Analogue-to-digital converter
ANSI	American National Standards Institute
ASCII	American Standards Committee for Information Interchange
CAD	Computer aided design
CSMP	Continuous System Modelling Program
CW	Clockwise
DAC	Digital-to-analogue converter
DAQ	Data acquisition
DC	Direct current
DOS	Disk operating system
DoF	Degrees of freedom
DP	Dynamic programming
e.m.f.	Electromotive force
FBM	Forward-backward method
GPSS	General-purpose simulation system
H-J-B	Hamiltonian-Jacobi-Bellman
HW	Hardware
IC	Integrated circuit
IP	Index of Performance
I/O	Input/output
L-E	Lagrange-Euler
LQR	Linear quadratic regulator
LTI	Linear time invariant
MATLAB	Matrix laboratory
MIMO	Multiple-input multiple-output
MIT	Massachusetts Institute of Technology
MP	Maximum principle
MRAC	Model-referenced adaptive control
N-E	Newton-Euler
NN	Neural network
PC	Personal computer
PCB	Printed circuit board
PD	Proportional and derivative
PID	Proportional, integral and derivative
PLC	Programmable logic controllers
PVC	Path velocity controller
PWM	Pulse width modulated
RAPL	Robotic Automatic Programming Language
RMAC	Resolved motion acceleration control
RMRC	Resolved motion rate control

Chapter 1

Introduction

1.1 Motivation

An industrial robot is defined by the U.S. Robot Industries Association as a *reprogrammable, multifunctional manipulator designed to move material, parts, tools, or specialised devices through variable programmed motions for the performance of a variety of tasks.* Similar definitions are adopted by the British Robot Association and the Japanese Robot Association [I]. In short, a robot is a reprogrammable general-purpose manipulator with external sensors that can perform various assembly tasks. With this definition, a robot must possess *intelligence*, which is normally due to computer algorithms associated with its control and sensing devices [2]. The first robotic patents were by G.C. Devol for parts transfer machines in the mid 1950's. Further development of this concept by G.C. Devol and J.F. Engelberger led to the first industrial robot, introduced by Unimation Inc., in 1959 [1].

When first introduced commercially, robotic manipulators were used in dedicated mass production activities such as automobile or printed circuit board assembly. These are volume manufacturing operations with economies of scale that justify a robot's high cost. Robots have also proved useful in performing tasks that require extraordinary strength, endurance, dexterity and for operating under conditions that might threaten the health of a human worker, such as nuclear environments. And, of course, for carrying out repetitive or monotonous tasks that are associated with high levels of fatigue, accidents and human error.

More recently, flexible automation systems [3] (those which are capable of producing a variety of products with virtually no time lost for changeovers from one product to the next) have extended the range of activities for which robots can be cost-effectively employed to a multitude of lower volume operations. Laboratory automation is one of these areas, where a manipulator designed to perform tasks such as sample preparation or drug analysis must incorporate many of the traits common to the most advanced industrial robots within the framework of an analytical laboratory, i.e.:

- Shorter turnaround time, resulting in overall cost reduction and increased productivity.
- The same or better precision and accuracy than existing manual methods with improved quality and reliability of measurement.
- The freeing of trained laboratory staff to do more creative and productive work.
- Reduced, As Low As Reasonably Practicable (ALARP [4]) human contact with biological or chemical hazards.
- Lower consumption of sample and/or reagents used in automated analysis.

An automated radiopharmaceutical dispenser [5] developed at Middlesex University [2] is a clear example of this new field of research which is, understandably, attracting the attention of industry. The system prepares precise individual patient prescriptions with the required dose of radio-isotope delivered either in shielded syringes or vials. This is automatically accomplished by the use of a

[1] These and other robotic-related resources can be found at the Frequently Asked Questions list for the internet robotics newsgroups comp.robotics.misc and comp.robotics.research, at `http://www.frc.ri.cmu.edu/robotics-faq`.

[2] In a collaborative link with British Nuclear Fuels Plc. and St. Bartholomew's Hospital (London).

number of computer controlled workstations around a general purpose manipulator at the heart of the system.

The origins of the work covered in this thesis stem from the broad objective of making this system operate in a more productive way according to the first of the aforementioned goals. Since the robot subsystem can be easily regarded as the key element to maintaining an operative and flexible robotic automated laboratory, the focus of this work has been on designing (and implementing) new control strategies that could improve the performance of the robotic arm. It is this general aim that spurred the author's interest in the theory of optimal control systems. As early as 1965 Dorf [6] wrote:

> *"It is the high-speed computer that has allowed the solution of optimal control problems. Perhaps, more realistically, it is the* expected *high-speed and large memory computers in the next generation of development and improved programming techniques that will allow control theorists to consider the solution of realistic problems."*

Optimal control theory applied to the area of robotics has been the subject of rather extensive research which dates back to the early 1970's when Kahn and Roth [7] published their paper. Despite the interest that the paper generated, very little effort has been dedicated to employing the "next generation" technology envisaged by Dorf to find practical solutions applicable to present industrial manipulators. The theory and experimental work developed in this dissertation is intended as a step forward in practical optimal control of the robot motion problem.

1.2 Overview - The Problem within its Context

One of the basic problems in robotics is planning motions to solve some specific task and then controlling the response of the robot to achieve those motions. It is common practice to refer to the `path` as the curve in space that the manipulator end-effector must follow during the motion with central attention on collision avoidance. The path is a time-invariant function of one parameter, S, the amount of the curve traversed. However, quite often the description of the desired path is simply provided in terms of the path's end-points and possibly a set of intermediate via points or corner points to avoid obstacles in the workspace. The `trajectory` is defined as the time sequence of intermediate configurations of the arm $q_i(t), i = 1, 2, \ldots n$ (where n represents the number of Degrees of Freedom - DoF - of the manipulator) along the programmed path. These configurations, possibly together with their first and second time derivatives, are then fed to the servo mechanisms `controlling` the actuators that actually move the arm. These steps, schematically depicted in Figure 1.1, are followed by most current industrial manipulators to accomplish a specified task because the overall motion process could become fairly complicated if considered in its entirety [8].

For current robots, path planning and programming is usually accomplished by human operators who rely on expensive, time-consuming and semi-empirical methods to construct the programs which will carry out the desired task. These include off-line programming, the use of a teach pendant, or manually leading the robot through the desired path first [9]. Considerable research has been reported into path planning techniques that can incrementally automate robot programming. The methods have the advantages of eliminating programming costs for new paths, reducing down-time and set-up time and allowing robots to be used for changing tasks in changing environments. Planning can be improved in several ways: the world models used in planning can be refined (with particular attention to the space used for their representation), optimal path planning methods for solving a given task can be generated, and the ability to specify the robot motions required to achieve a task in terms of high-level task commands can be investigated. However, much of the research in this area has been devoted to devising methods to automatically plan collision-free paths, which has led to two competing techniques. One of the methods solves the problem by forming a connected graph representation of the free space and then searching the graph for a collision-free path [10]. Unfortunately this technique has exponential complexity in the number of joints in the device. A second approach is based on creating artificial potential fields around obstacles which cause the manipulator(s) to avoid the obstacles while they are drawn towards an artificial attractive pole at the goal point [11]. Unfortunately, this method generally adopts a local view of the environment and is subject to the manipulator becoming stuck at local minima of the artificial field. As a result, systems that plan collision-free paths are not as yet available commercially. In general, path planning processes are normally independent of improving

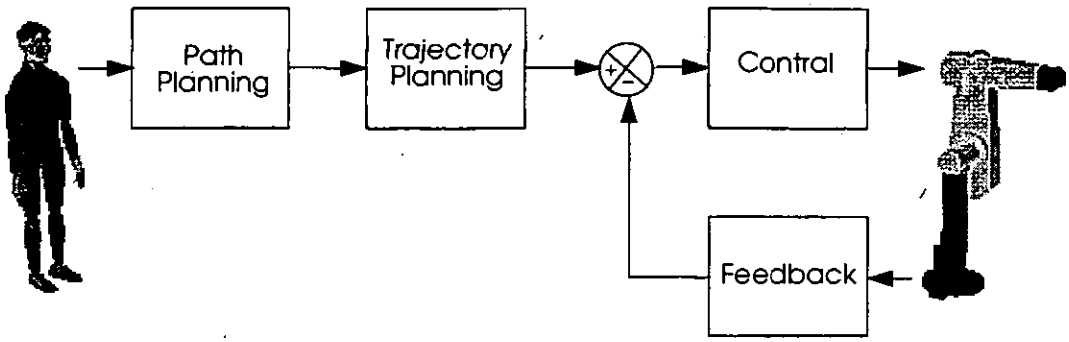

Figure 1.1: Traditional manipulator motion block diagram.

execution time of a given path and will not be addressed in this dissertation. For a good review of recent developments in the area of automatic programming and motion planning the reader may refer to [9].

As will be examined in later Chapters, traditional trajectory planning strategies such as low degree polynomials, cubic splines, linear functions with parabolic blends, critically damped or bang-bang [12] perform some sort of interpolation between control set points to obtain the time course along the desired path. These are entirely based on kinematic considerations to satisfy a specified set of constraints on the position, velocity and/or acceleration of the arm along the path. These algorithms are meant to be efficient in the sense of minimising computational expense. Hence, simplified constraints are often assumed along the segments that define the path [13, 14].

Practically all industrial manipulators currently in use are based on classical linear feedback controllers with proportional and derivative (PD) or proportional, derivative and integral (PID) algorithms [15] to track the desired trajectory. However, it is well known that mechanical manipulators are multibody systems whose dynamic behaviour is described by strongly non-linear differential equations [16]. Non-linearities are associated both with position and velocity variables, and also payloads. While a few advanced model-based industrial controllers [12, 17] compensate for some of the position-dependent non-linear terms, such as gravity, they very often neglect the velocity-dependent terms in the controller design. Although industrial trackers can generally keep the manipulator fairly close to the desired trajectory [18], this simplistic division of robot motion into trajectory planning and tracking often results in mathematically tractable solutions which do not utilise the manipulator's full capabilities. The source of such underutilisation lies in the fact that both maximum speeds and accelerations/decelerations are limited for a given robot structure by the torque capacity of the joint actuators which vary across the workspace. Yet when trajectories are planned, constant maximum bounds along each DoF are assumed [13, 14]. As a consequence, these specifications must he chosen conservatively in order not to exceed the actual capabilities of the device, possibly forcing the robot to be underutilised [19].

To overcome these drawbacks, the trajectory planning problem is often reformulated as an optimal control problem with state and control constraints, thus taking into account the dynamic characteristics of the manipulator in the optimization procedure [20]. In doing this, the traditional inefficient division of trajectory planning and control as two separate motion stages is removed. The resulting algorithms yield optimal/suboptimal trajectories with respect to some performance index (e.g. time, control action, acceleration) along with an approximation to the open-loop optimal/suboptimal control torques that would generate such trajectories. However, they all suffer from the same shortcomings: they result in impractically complicated schemes, most of them solved via iterative numerical algorithms, hence at a large computational expense. In fact, results to date have been demonstrated only in simulations and very few authors have discussed implementation issues or presented experimental results. Exceptions are [21, 22, 23] who demonstrated the merit of time optimal control, and showed the significant contribution that the often ignored motor dynamics have in the optimization process. However, practical limitations have restricted the work to motion along specified paths and as stated in [24], to date, no practical method has been developed for the on-line time-optimal feedback control of manipulators with non-linear dynamics.

1.2.1 The novel approach

The aim of the work presented in this investigation is to go beyond computing a trajectory that is optimal with respect to a timing performance index and to study the problem in a manner that allows for an on-line, practical implementation of the solution. With this goal in mind, the approach taken departs from the usual solutions (described later in this work) which either assume an initial given path, or otherwise fit the optimal solution to a piecewise polynomial function. In contrast, the work developed here is based upon the fact that many robotic applications do not require manipulators to strictly follow a prescribed trajectory. This is particular true when specifying *gross motion* of the robot arm when it operates in a collision-free space [3]. Hence, the manipulator control problem can be formulated in a more general form in which the robot is given freedom to move along any trajectory between any two given intermediate or end-points. "Pick and place" routines in most laboratory automated environments fall within this group. The latter can usually be described as highly compact enclosures where a large number of via points are necessary to avoid collision with obstacles during the execution of a particular task. Yet an effort can be made to design trajectory planners between set points to operate the manipulators at their maximum efficiency.

A practical algorithm for the on-line point-to-point unconstrained optimal motion of robot manipulators is proposed in this investigation which results in an improvement over the commercial joint-interpolated approach and does not need pre-programming of the trajectory. The proposed scheme, which takes the full rigid-body dynamics of the manipulator into account, results in a two-point boundary-value (TPBV) problem in joint space which is analytically solved in real-time using Pontryagin's Maximum Principle (MP) assuming bang-bang control for each robot joint. Unfortunately, the solution has not been achieved without a compromise. The difficulty of analysing non-linear systems led to the need to locally linearise the dynamic model to avoid an iterative and computationally expensive solution. This was a deliberate choice that restricted the solution to near-time-optimal control to enable a practical implementation. However, it will be shown how this mathematical shortcoming with regards to optimality did not significantly degrade the performance of the solution under the normal operating conditions of the manipulator. The idea of the *averaged dynamics*, first proposed by Kim and Shin [25], was adopted in this investigation because it provides a twofold advantageous mechanism for the purpose of this dissertation:

1. It is a simple solution to the piecewise linearisation of the complex robot dynamics on the basis of current and goal state, hence suitable for a real-time solution.

2. It results in a feedback form controller, and is therefore appropriate for direct implementation without the need for a secondary trajectory tracking controller; moreover, the feedback structure also accounts for modelling errors and the implicit errors due to the dynamics approximation.

With the ultimate goal in mind of implementing the proposed approach in a real industrial manipulator, to some extent the most salient feature of this work, some other unique features of the investigation undertaken here are stated below:

- The consideration of the manipulator electro-mechanical characteristics in the overall design of the optimal controller, in particular during the testing of the algorithm.

- A refined approach to the representation of the overall dynamic behaviour of the manipulator during motion. This is accomplished by the use of a weighting factors for the dynamic performance of each boundary condition.

- An investigation into the stability of the scheme.

1.2.2 Aims and objectives

The aims of this work are to investigate and develop a feasible time-optimal control algorithm in feedback form for the general point-to-point unconstrained motion of robot manipulators. Research objectives are:

- Review of the published literature describing previous relevant contributions.

[3] Otherwise, some sort of collision avoidance can be assumed at task level to specify appropriate collision-free control points.

- Identification and development of the dynamic model characteristics (including the electro-mechanical parts) of the industrial manipulator employed in this work (CRS A251).
- Development and simulation of an optimal control strategy in an advanced graphical simulation environment (Deneb's TELEGRIP).
- Design and implementation of hardware necessary to interface the master controller implemented on a PC with the industrial robot controller.
- Controller performance testing on the real system.
- Comparison and analysis of the results.
- Recommendations for further work.

1.2.3 Statement of originality

The material in this thesis is the original work of the author. Contributions made by undergraduate students were under the supervision of the author, who conceived and proposed the project concepts.

1.3 Outline

The general outline of the contents of this dissertation is as follows:

Chapter 2 reviews basic strategies widely applied to the description of manipulator motion in terms of trajectories through space.

Chapter 3 is concerned with methods of controlling the robot arm so that it tracks the desired trajectories or follows a prescribed path through space. An exhaustive examination is not aimed for. Instead, the emphasis is on placing optimal control within the context of general control strategies, and on understanding their basic principles by reference to specialised literature where detailed information can be obtained.

Chapter 4 is a discussion of the relevant research literature on optimal trajectory planning and control of robot manipulators, mainly focused on time-optimal strategies. An ad-hoc classification based on whether the manipulator is constrained to move along a specified path or not, gives rise to the fundamental nature of the alternative proposed in this work. Special emphasis is also placed on reviewing optimal robot control approaches which are feasible for implementation on-line.

Chapter 5 deals with the derivation and validation of the electro-mechanical manipulator equations of motion, based on a Lagrangian mechanics point of view.

Chapter 6 covers the analysis and design of the (near) optimal control strategy, which is based on Pontryagin's MP. It is shown how the proposed strategy can be seen as an extension of the fundamentally simpler "double integrator" problem, and an investigation of the assumptions needed to make the method viable for an on-line implementation is treated in depth. Extended considerations about the stability of the controller are also discussed.

Chapter 7 describes the (graphical) simulation environment where the control strategy was first tested, and the test-rig developed to implement the controller. Results obtained from the real system in moving the manipulator between a large number of configurations in the workspace are presented and compared with the widely employed PID linear control of industrial manipulators. It is shown how the proposed (near) time-optimal approach can be used as a planning and control tool to determine fast robot movements with good dynamic properties, demonstrating specific improvements over more traditional linear motion schemes.

Chapter 8 summarises the work presented in this thesis, and draws some conclussions about the practicalities of the proposed solution to real robot manipulators. Furthermore, a number of ideas are suggested to improve the strategy presented.

Appendix A presents the derivation of the manipulator equations of motions, deferred from the main body of text.

Appendix B provides the source code for the time optimal controller implemented on the test-rig.

Appendix C shows the circuit diagram and schematics of the interface printed circuit board (PCB) designed for the implementation of the new control strategy.

Appendix D displays some of the reference polynomial trajectories employed during the validation of the manipulator dynamic model in Chapter 5.

Appendix E corresponds to the video included with this thesis, which shows the real-time and simulation setup described in this work.

Appendix F is a collection of the papers published so far as a result of this work.

References

[1] Rivin E.I. *Mechanical Design of Robots.* McGraw-Hill Book Co., (1988).

[2] Fu K.S., Gonzalez R.C., and Lee C.S.G. *Robotics: control, sensing, vision, and intelligence.* McGraw-Hill Book Co., (1987).

[3] Schoeny D.E. and Rollheiser J.J. The automated analytical laboratory: Introduction of a new approach to laboratory robotics. *American Laboratory*, pages 42–47, September (1991).

[4] Elliot A.T., Murray T., and Hilditch T.E. Automated radiopharmaceutical dispensing. *Nuclear Medicine Communications*, **9**:363–367, (1988).

[5] Stoker M., Solanki K., Gill R., Hardie R., and Snoback R. Hamilton dispenser - a complete automated radiopharmaceutical dispenser (HAPD). In *European Association of Nuclear Medicine Congress Proceedings*, Laussane, Switzerland, (1993).

[6] Dorf R.C. *Modern Control Systems.* Addison-Wesley Publishing Company Inc., fourth edition, (1986).

[7] Kahn M.E. and Roth B. The near-minimum-time control of open-loop articulated kinematic chains. *Journal of Dynamic Systems, Measurement, and Control. Transactions of the ASME*, **93**(3):164–172, September (1971).

[8] Brady M., Hollerbach J.M., Johnson T.L., Lozano-Perez T., and Mason M.T. *Robot Motion: planning and control.* MIT Press, (1982).

[9] Sanders D.A. *Making Complex Machinery Move - automatic programming and motion planning.* Robotics and Mechatronics Series. Research Studies Press Ltd., (1993).

[10] Lozano Perez T. A simple motion planning algorithm for general robot manipulators. *IEEE Journal of Robotics and Automation*, **RA-3**(3):224–238, June (1987).

[11] Khatib O. Real time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, **5**(1):90–99, Spring (1986).

[12] Craig J.J. *Introduction to Robotics: mechanics and control.* Addison-Wesley Publishing Company Inc., second edition, (1989).

[13] Luh J.Y.S. and Lin C.S. Optimum path planning for mechanical manipulators. *Journal of Dynamic Systems, Measurement, and Control. Transactions of the ASME*, **103**:142–151, June (1981).

[14] Lin C.S., Chang P.R., and Luh J.Y.S. Formulation and optimization of cubic polynomial joint trajectories for industrial robots. *IEEE Transactions on Automatic Control*, **AC-28**(12):1066–1073, December (1983).

[15] Franklin G.F., Powell J.D., and Emami-Naeini A. *Feedback Control of Dynamic Systems.* Addison-Wesley Publishing Company Inc., (1986).

[16] Faessler H. Robot control in cartesian space with adaptive non-linear compensation. In Schweitzer G. and Mansour M., editors, *Dynamics of Controlled Mechanical Systems*. Springer-Verlag, (1988).

[17] An C.H., Atkeson C.G., and Hollerbach J.M. *Model-Based Control of a Robot Manipulator*. MIT Press, (1988).

[18] Shin K.G. and McKay N.D. Minimum-time control of robotic manipulators with geometric path constraints. *IEEE Transactions on Automatic Control*, **AC-30**(6):531–541, June (1985).

[19] Kim B.K. and Shin K.G. Minimum-time path planning for robot arms and their dynamics. *IEEE Transactions on Systems, Man and Cybernetics*, **SMC-15**(2):213–223, March/April (1985).

[20] Gilbert E.G. and Johnson D.W. Distance functions and their application to robot path planning in the presence of obstacles. *IEEE Journal of Robotics and Automation*, **RA-1**(1):21–30, March (1985).

[21] Wapenhans H., Hölzl J., Steinle J, and Pfeiffer F. Optimal trajectory planning with application to industrial robots. *The International Journal of Advanced Manufacturing Technology*, **9**:49–55, (1994).

[22] Shiller Z. Time-energy optimal control of articulated systems with geometric path constraints. *Journal of Dynamic Systems, Measurement, and Control. Transactions of the ASME*, **118**:139–143, March (1996).

[23] Shiller Z., Chang H., and Wong V. The practical implementation of time-optimal control for robotic manipulators. *Journal of Robotics & Computer-Integrated Manufacturing*, **12**(1):29–39, (1996).

[24] Sundar S. and Shiller Z. A generalized sufficient condition for time-optimal control. *Journal of Dynamic Systems, Measurement, and Control. Transactions of the ASME*, **118**:393–396, June (1996).

[25] Kim B.K. and Shin K.G. Suboptimal control of industrial manipulators with a weighted minimum time-fuel criterion. *IEEE Transactions on Automatic Control*, **AC-30**(1):1–10, January (1985).

Chapter 2

Fundamentals of Robot Trajectory Planning

2.1 Introduction

This Chapter will examine basic obstacle-free trajectory planning strategies widely used on most current industrial manipulators. It is not self-contained, as it is assumed that the reader will be familiar with most of the mathematics and terminology. General references are cited which contain fuller expositions on these topics. In Section 2.2 the formalisms of the two usual approaches for specifying and planning trajectories, i.e., joint interpolated and Cartesian space, are discussed in the framework of the work presented here. The fundamental properties of the most common parameterised joint-interpolated trajectories are reviewed in Sections 2.3–2.7, each Section building on the foundations laid in the previous one. Finally, a discussion of the deficiencies of the approaches described is presented in Section 2.8, to provide the necessary background for understanding the solutions proposed by other researchers, surveyed in Chapter 4, and also the approach presented in this work.

2.2 General Considerations on Trajectory Planning

Trajectory planning schemes generally interpolate or approximate the desired path and generate a sequence of time-based control set-points for the control of the manipulator from the initial location to its destination. Path end-points can be specified either in joint coordinates or in Cartesian coordinates. However, they are usually specified in Cartesian coordinates because it is easier for the user to visualize the correct end-effector configuration. If, on the other hand, the trajectory is to be generated from constraints on the path specified in joint coordinates, then the inverse kinematics transformation must be computed first. This is customarily an ill-defined non-linear transformation because of its one-to-many mapping characteristics. Thus, except for manipulators with a proper kinematic structure, it can not be expressed analytically and is, in general, problematic.

Two common approaches are then used to actually generate the manipulator trajectory which are also based on the distinction between joint and Cartesian space.

1. In the first of these approaches, the user *implicitly* describes the path and trajectory followed by the arm by specifying, in joint coordinates, a set of constraints on its position, velocity and acceleration at selected locations (called knot, via, or interpolation points) along the desired trajectory. The trajectory planner then chooses one of a class of parameterised trajectories (described below) that satisfies these constraints.

2. In the second approach, the user *explicitly* specifies the path that the manipulator must traverse by an analytical function, such as a straight line or a circular path in Cartesian coordinates, and the trajectory planner determines a desired trajectory either in joint or Cartesian coordinates that approximates the desired path.

In the first approach, the constraint specification and the planning of the manipulator trajectory are performed in joint coordinates (the former might be the result of the inverse kinematic conversion from

some specified Cartesian points as mentioned above). In the second approach, path constraints and the path itself are specified in Cartesian coordinates, but because servoing is performed in the joint-variable space, Cartesian positions must be converted into their corresponding joint solutions. This is generally accomplished in two steps. First, a number of knot points or intermediate configurations in Cartesian coordinates along the Cartesian path are selected. The path segments defined by two adjacent knot points are then approximated by specifying a class of functions according to some criteria. To achieve the latter step, two major approaches emerge: the Cartesian space-oriented method and the joint-space method. In the former approach, most of the computation is performed in Cartesian coordinates, which are then converted to their corresponding joint solutions by the inverse kinematic transformation. The resulting trajectory is a piecewise straight line between adjacent knot points and the subsequent control is performed at the hand (end-effector) level [1]. In the joint-space oriented method, a low-degree polynomial function in joint space is employed to approximate the Cartesian path segment bounded by the two adjacent Cartesian knot points (which are initially transformed into joint displacements). The resultant Cartesian trajectory is not a piecewise straight line as before, and the resultant control is completed at the joint level [2].

The Cartesian space-oriented approach has the advantage of being a straightforward concept, and a certain degree of accuracy is assured if the desired path is a straight line. Moreover, it is easier to determine the locations of the various links and the hand during motion, a task that is usually required to guarantee obstacle avoidance and physical realisation along the trajectory. On the other hand, trajectory planning in the joint-variable space (either directly or as an approximation to Cartesian paths) has a number of advantages:

- The trajectory is planned directly in terms of the controlled variables during motions.
- Trajectory planning can be performed in real-time since the inverse kinematic solution routine and the Jacobian transformations do not have to be called upon to make the conversion at each control point along the trajectory, as would be the case for Cartesian space-oriented methods.
- For this same reason, degeneracies such as position redundancies and velocity singularities of the manipulator are not a problem.
- Joint trajectories are easier to plan.

There is yet another advantage in using joint interpolated trajectories that will be readily apparent later in the dissertation when manipulator dynamic constraints are introduced. However, with regard to the choice of the coordinate system, it can be noted now that physical constraints like actuator torque/force, velocity and acceleration are bounded by joint coordinates. Thus, in adopting Cartesian based trajectory strategies the resulting problem would have mixed constraints in two different coordinate systems, an undesirable characteristic from the efficiency point of view. Because of the various advantages mentioned above, trajectory planning in joint coordinates is the widely used strategy in current manipulator control and the work described in this thesis will be restricted to the study of trajectories in the joint space. Further reading about the topic of Cartesian trajectories can be found in [1, 3, 4].

There is a general consensus about what criteria can be identified for evaluating trajectories and trajectory planners, and most authors presume the following characteristics for a well posed trajectory [5]:

- Efficiency, both to compute and execute.
- Accuracy and predictability. Trajectories should not degenerate unacceptably near a singularity.
- It is also generally desirable to design the trajectory as a smooth function of time, i.e., one which is continuous and ideally has a continuous first derivative.

The trajectories reviewed in the following Sections fulfill most or all of these characteristics, and also represent strategies implemented in most current industrial manipulators. Some authors [4, 6, 7] have also suggested a continuous second derivative since high jerk motions tend to cause increased wear on the mechanism and cause vibration by exciting resonances in the manipulator. An introduction to one such class of trajectories is presented in Section 2.7 where spline trajectories are introduced. On the other hand, non-continuous acceleration is sometimes cited [8] as a desirable alternative since it can lead to minimum-time trajectories as further described in Section 2.6 below. This generic form of trajectory

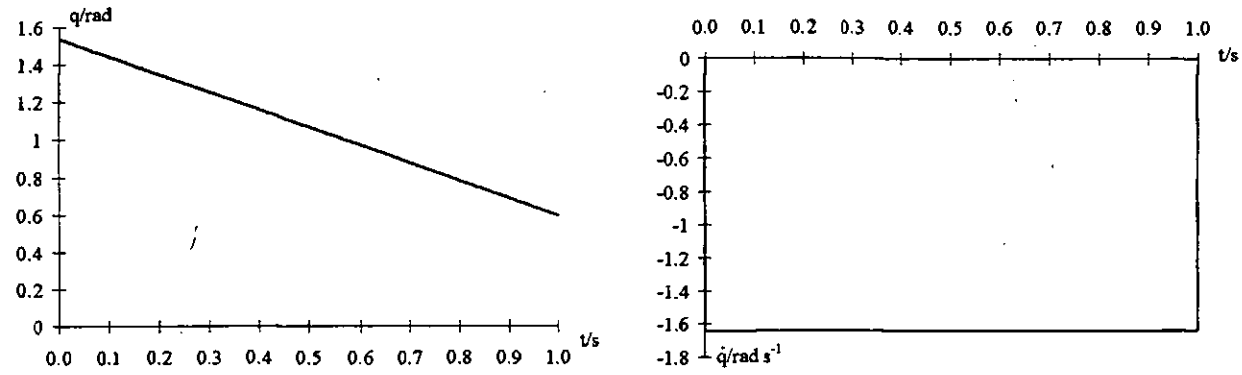

Figure 2.1: Position (left) and velocity for first order polynomial trajectories.

is of great importance in the work undertaken in this investigation, as will be readily apparent in future Chapters.

2.3 First Order Polynomials

The simplest description of a trajectory is a parametric specification of the initial (t_i) and final (t_f) position constraints $q(t_i)$ and $q(t_f)$ where q is the joint generalized coordinate, which can be satisfied by a trajectory of the form

$$q(t) = f(t)q(t_f) + (1 - f(t))q(t_i) \tag{2.1}$$

where $f : [0,1] \Longrightarrow [0,1]$ is any continuous function satisfying $f(0) = 0, f(1) = 1$. The simplest such function is $f(t) = t$ in which case $q(t)$ is a linear polynomial combination of the end-points $q(t_i)$ and $q(t_f)$, i.e.,

$$q(t) = q(t_i) + (q(t_f) - q(t_i))t \tag{2.2}$$

Although simple, the joint velocity $\dot{q}(t)$ is constant throughout the motion (namely $\dot{q}(t) = q(t_i) - q(t_f)$), as seen in Figure 2.1, and therefore acceleration $\ddot{q}(t) = \pm\infty$ is required at the boundary times, t_i and t_f, to satisfy this constraint. Although mathematically correct, this is not physically attainable. Yet another drawback is that there is no guarantee that the joint solution $q(t)$ always lies in the workspace.

2.4 Cubic Polynomial

Suppose the trajectory is further constrained by specifying its initial and final velocities $\dot{q}(t_i)$ and $\dot{q}(t_f)$, satisfying the continuity in velocity. The four constraints on position and velocity can now be satisfied by the following cubic polynomial trajectory

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{2.3}$$

Assuming time is normalized between $[0,1]$, coefficients can be easily derived from the constraints as

$$\begin{aligned} a_0 &= q(t_i) \\ a_1 &= \dot{q}(t_i) \\ a_2 &= -3q(t_i) - 2\dot{q}(t_i) + 3q(t_f) - \dot{q}(t_f) \\ a_3 &= 2q(t_i) + \dot{q}(t_i) - 2q(t_f) + \dot{q}(t_f) \end{aligned} \tag{2.4}$$

Furthermore, in point-to-point motions initial and final velocity conditions are null, so that the expression above can be simplified as

$$\begin{aligned} a_0 &= q(t_i) \\ a_1 &= 0 \\ a_2 &= 3(q(t_f) - q(t_i)) \\ a_3 &= -2(q(t_f) - q(t_i)) \end{aligned} \tag{2.5}$$

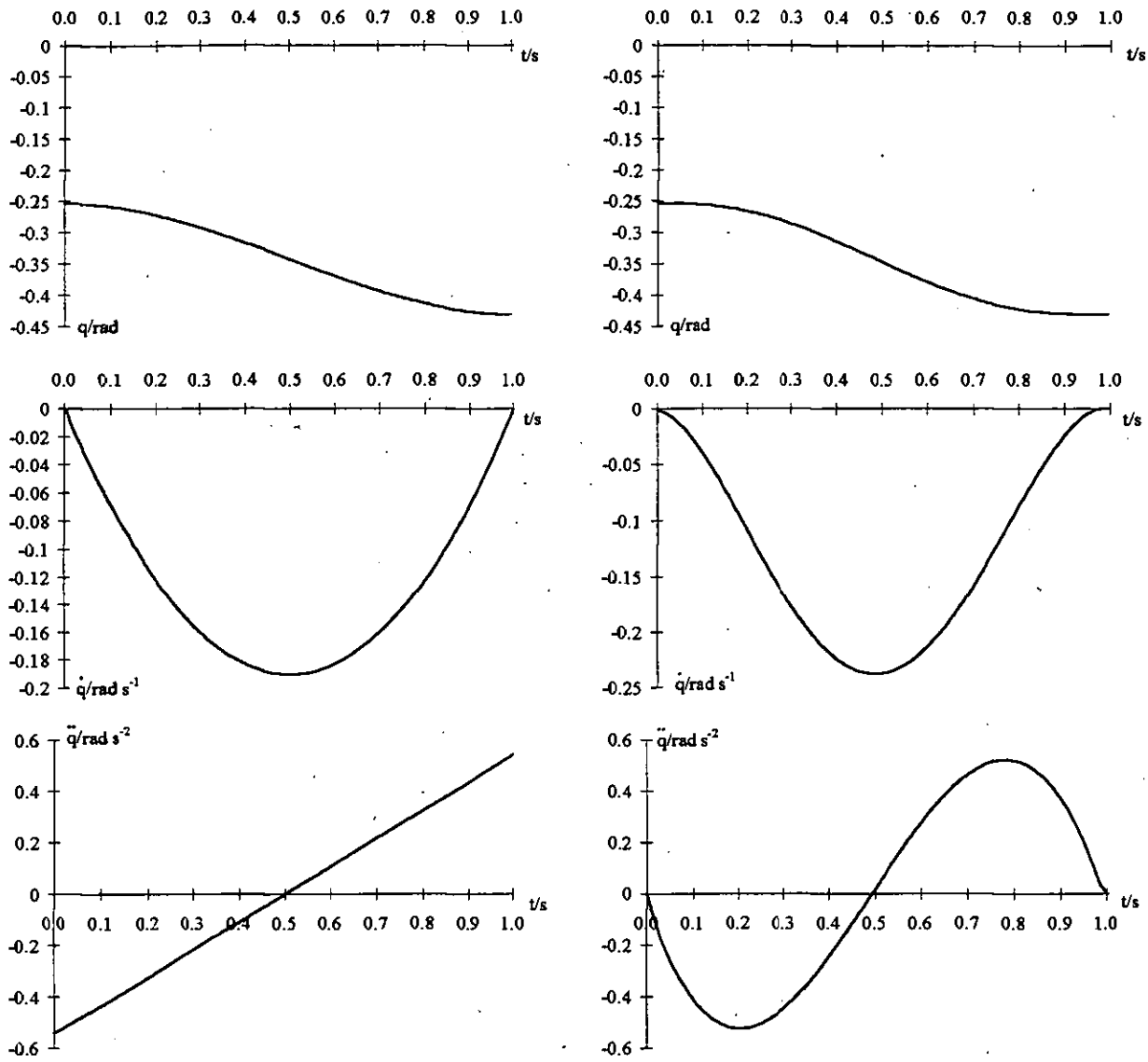

Figure 2.2: Position (top), velocity (middle) and acceleration for third (left) and fifth (right) order polynomial trajectories.

Just as the linear function that satisfies positional constraints, the cubic polynomial is an over-simplification that does not take into account the fact that there is a maximum attainable velocity. Furthermore, acceleration is a discontinuous function that grows linearly with time. This can be easily seen in the acceleration profile of Figure 2.2 (left), obtained by simulating trajectory (2.3) with the coefficients given by (2.5).

2.5 Quintic Polynomials

If it is desired to specify the acceleration at both ends of the trajectory as well as the velocity and position, there are a total of six constraints to be met. A quintic polynomial suffices

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \tag{2.6}$$

Working with the same assumptions as those above in (2.5) and also initial and final null acceleration

the following coefficients are obtained

$$\begin{aligned} a_0 &= q(t_i) \\ a_1 &= 0 \\ a_2 &= 0 \\ a_3 &= 10(q(t_f) - q(t_i)) \\ a_4 &= -15(q(t_f) - q(t_i)) \\ a_5 &= 6(q(t_f) - q(t_i)) \end{aligned} \tag{2.7}$$

Figure 2.2 (right), shows this trajectory obtained at the same sample rate as for the cubic trajectory on the left. Note that the acceleration profile is no longer linear, adding continuity to the overall trajectory.

2.6 Other Primitive Polynomials

Besides the simple function $f(t) = t$, other bases can be chosen that satisfy the continuity conditions expressed in Section 2.3, for example a function of the form $f(t) = \cos(\pi/2(1-t))$. The initial and final position and velocity constraints satisfied by the cubic polynomial given in (2.4) are also achieved by the following cosine trajectory

$$q(t) = \cos^2(\frac{\pi}{2}(1-t))\{q(t_f) - \frac{2\dot{q}(t_f)}{\pi}\cos(\frac{\pi t}{2})\} + \cos^2(\frac{\pi t}{2})\{q(t_i) + \frac{2\dot{q}(t_i)}{\pi}\cos(\frac{\pi}{2}(1-t))\} \tag{2.8}$$

Restricting attention to the point-to-point motion constraints of null initial and final velocity, plus initial and final acceleration that is either null or of maximum value, a comparison of several polynomial trajectories is illustrated in Figure 2.3. The parameters can be chosen so that the resulting trajectories can be reasonably similar [8]. Of particular attention for future reference in this report is the bang-bang trajectory, consisting of a period in which the maximum acceleration is applied to move the arm from rest, followed by an equal period of maximum deceleration to stop it.

$$q(t) = \begin{cases} q(ti) + (\ddot{q}_{max}/2)t^2 & \text{if } 0 \le t \le 1/2 \\ q(tf) - (\ddot{q}_{max}/2)(1-t)^2 & \text{if } 1/2 \le t \le 1 \end{cases} \tag{2.9}$$

Mutjaba [8] found that the trajectories based on a fifth degree polynomial, a cosine, and a sine added to a linear ramp are about 10-20% slower than the bang-bang trajectory. It was also observed that the critically damped trajectory - consisting of a sum of decaying exponentials of the form e^{at} - is the principal mode of operation of a large number of industrial manipulators, yet it was estimated to be about three times slower than the other trajectories, as shown in Figure 2.3.

2.7 Splined Low-order Polynomials

The polynomial interpolation schemes proposed so far enable the manipulator to follow trajectories which are smooth functions of time. However, a trajectory description may include constraints other than those that derive from the initial and final configurations of the arm as shown in Figure 2.4. In particular, knot points could be added to the terminal constraints to prevent the arm colliding with an object in the workspace. Complete coverage is beyond the goal of this thesis and will not be considered any further. Furthermore, as shown in Figure 2.4, knot points could also represent important trajectory configurations that would guarantee admissible departure and approach directions, or contribute to drive the manipulator at the *extrema* conditions, i.e., maximum recommended speed and/or acceleration.

By continuing the development of previous Sections, it is always possible to satisfy an arbitrary number of constraints by a polynomial trajectory of sufficiently high degree [5]. However, a number of drawbacks to this approach have been stated in the literature:

- The difficulty of checking a polynomial trajectory of degree n for violation of physical constraints (i.e., positions within the workspace) during an arm movement increases rapidly with n.
- It has been suggested [9] that high degree polynomials "have an unfortunate tendency to overshoot and wander".

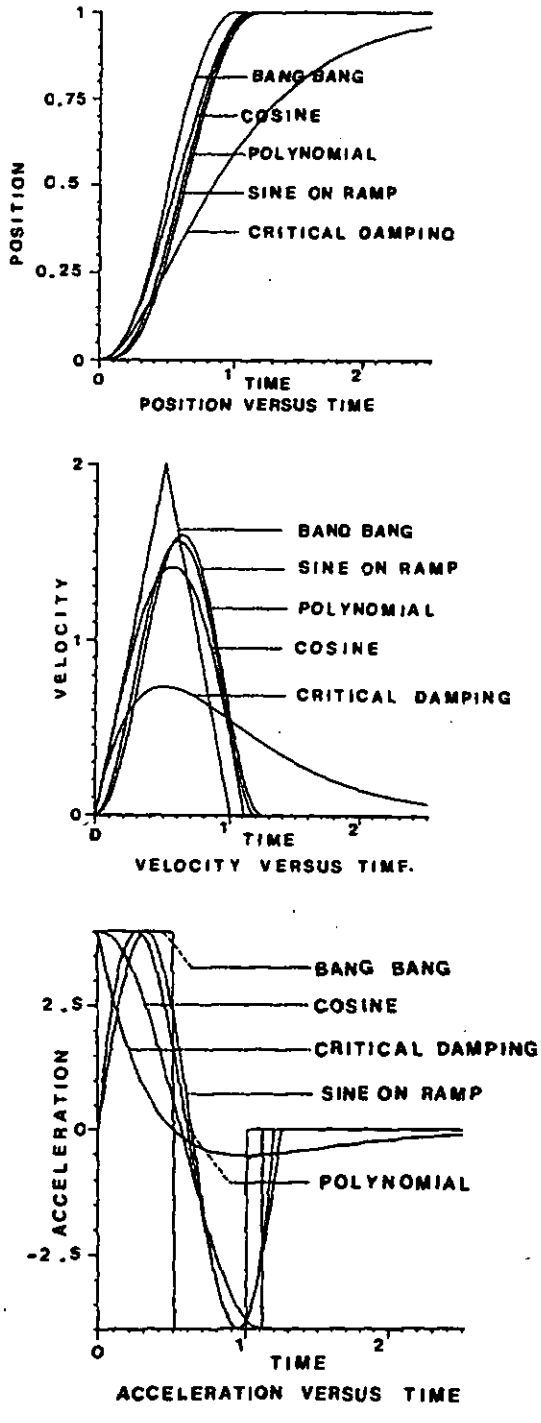

Figure 2.3: Position, velocity and acceleration for different polynomial trajectories [5].

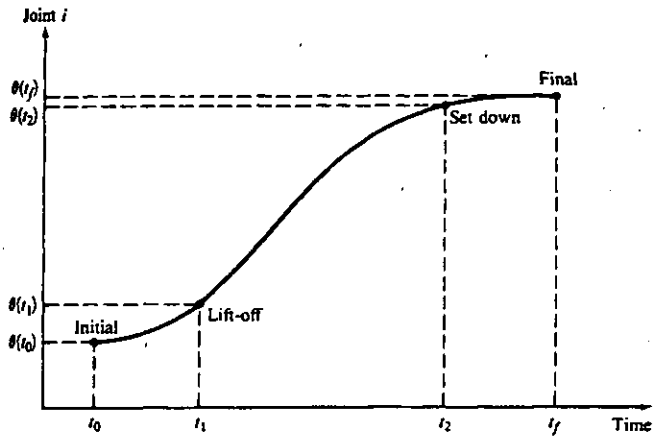

Figure 2.4: Point-to-point trajectory constrained by 2 knot points [3].

- The numerical accuracy to which a polynomial can be computed decreases as n increases.
- It is expensive to compute the parameters of polynomial trajectories.

An alternative approach is to split the entire joint trajectory into several trajectory segments running hetween knot points. Different primitive interpolating polynomials of a lower degree, like those described in previous Sections, can then he used to spline the trajectory segments together.

Definition 2.1 *A spline function of degree n, with knot points* $t_0 < t_1 < ... < t_m$, *is an (n-1)-time differentiable function f(t) which is equal on each interval* $[t_i, t_{i+1}]$ *between knot points to a polynomial of degree less thon or equal to n* [10].

According to this definition, the most common methods employed to spline a joint trajectory and guarantee continuity of position, velocity and acceleration at the polynomial boundaries use a minimum of third-degree polynomials. In general [3], two knot points may be specified as indicated in Figure 2.4: one near the initial position (lift-off) for departure and the other near the final position (set-down) for arrival which will contribute to safe departure and approach from/to the end-points. Each knot point imposes four constraints: two position constraints, as each of the splines is required to pass through the consecutive knot points, and two constraints to guarantee continuity of velocity and acceleration (or specify initial/final velocity and acceleration conditions in the initial/final points). The approaches to splined trajectory planning described next are, for the most part, equivalent in the sense that all satisfy the continuity constraint. However, depending on how these constraints are met, different trajectories arise:

2.7.1 3-5-3 spline

Four constraints can be satisfied by a cubic, so cubics can he used as the primitive trajectories in the first and last segment of the move. The mid-trajectory segment is a fifth-degree polynomial specifying the trajectory from the first to the second knot point which satisfies the six continuity constraints

imposed upon them. The constraints can be summarised as:

$$
\left\{
\begin{array}{l}
\text{First } 3rd \text{ degree polynomial} \\
\quad \text{constraints} = \left\{
\begin{array}{l}
\text{Initial position, } q(t_i) \\
\text{Initial velocity, } \dot{q}(t_i) \text{ (normally zero)} \\
\text{Initial acceleration, } \ddot{q}(t_i) \text{ (normally zero)} \\
\text{First knot point position , } q(t_1)
\end{array}
\right. \\
\text{Mid } 5th \text{ degree polynomial} \\
\quad \text{constraints} = \left\{
\begin{array}{l}
\text{Continuity in position at } t_1,\ q(t_1^-) = q(t_1^+) \\
\text{Continuity in velocity at } t_1,\ \dot{q}(t_1^-) = \dot{q}(t_1^+) \\
\text{Continuity in acceleration at } t_1,\ \ddot{q}(t_1^-) = \ddot{q}(t_1^+) \\
\text{Continuity in position at } t_2,\ q(t_2^-) = q(t_2^+) \\
\text{Continuity in velocity at } t_2,\ \dot{q}(t_2^-) = \dot{q}(t_2^+) \\
\text{Continuity in acceleration at } t_2,\ \ddot{q}(t_2^-) = \ddot{q}(t_2^+)
\end{array}
\right. \\
\text{Last } 3rd \text{ degree polynomial} \\
\quad \text{constraints} = \left\{
\begin{array}{l}
\text{Last knot point position , } q(t_2) \\
\text{Final position, } q(t_f) \\
\text{Final velocity, } \dot{q}(t_f) \text{ (normally zero)} \\
\text{Final acceleration, } \ddot{q}(t_f) \text{ (normally zero)}
\end{array}
\right.
\end{array}
\right. \tag{2.10}
$$

where t_1 and t_2 represent the time to arrive at the first and last knot points respectively.

2.7.2 4-3-4 spline

This trajectory, proposed in [11], relaxes the requirement that the interpolating polynomial must pass through the knot points exactly. Only the velocity and acceleration continuity constraints at the knot points are imposed. The new boundary conditions that this set of joint trajectory segment polynomials must now satisfy are:

$$
\left\{
\begin{array}{l}
\text{First } 4th \text{ degree polynomial} \\
\quad \text{constraints} = \left\{
\begin{array}{l}
\text{Initial position, } q(t_i) \\
\text{Initial velocity, } \dot{q}(t_i) \text{ (normally zero)} \\
\text{Initial acceleration, } \ddot{q}(t_i) \text{ (normally zero)} \\
\text{Continuity in velocity at } t_1,\ \dot{q}(t_1^-) = \dot{q}(t_1^+) \\
\text{Continuity in acceleration at } t_1,\ \ddot{q}(t_1^-) = \ddot{q}(t_1^+)
\end{array}
\right. \\
\text{Mid } 3rd \text{ degree polynomial} \\
\quad \text{constraints} = \left\{
\begin{array}{l}
\text{Continuity in velocity at } t_1,\ \dot{q}(t_1^-) = \dot{q}(t_1^+) \\
\text{Continuity in acceleration at } t_1,\ \ddot{q}(t_1^-) = \ddot{q}(t_1^+) \\
\text{Continuity in velocity at } t_2,\ \dot{q}(t_2^-) = \dot{q}(t_2^+) \\
\text{Continuity in acceleration at } t_2,\ \ddot{q}(t_2^-) = \ddot{q}(t_2^+)
\end{array}
\right. \\
\text{Last } 4th \text{ degree polynomial} \\
\quad \text{constraints} = \left\{
\begin{array}{l}
\text{Final position, } q(t_f) \\
\text{Final velocity, } \dot{q}(t_f) \text{ (normally zero)} \\
\text{Final acceleration, } \ddot{q}(t_f) \text{ (normally zero)} \\
\text{Continuity in velocity at } t_2,\ \dot{q}(t_2^-) = \dot{q}(t_2^+) \\
\text{Continuity in acceleration at } t_2,\ \ddot{q}(t_2^-) = \ddot{q}(t_2^+)
\end{array}
\right.
\end{array}
\right. \tag{2.11}
$$

where, as before, t_1 and t_2 represent the time to reach the first and last knot points respectively. It can be seen that the trajectory is now formed out of three trajectory segments: the first segment is a fourth-degree polynomial specifying the trajectory from the initial position to an initial knot point. The mid-trajectory segment is a third-degree polynomial from the first knot point to the second, whilst the last trajectory segment is a fourth-degree polynomial specifying the trajectory from the last knot point to the final position.

2.7.3 5-cubics spline

An alternative approach has been used at Stanford University [9]. Further knot points are introduced into the motion as "virtual" knot points. Unlike the two already specified which mark important intermediate configurations along the trajectory, the extra knot points are for mathematical convenience.

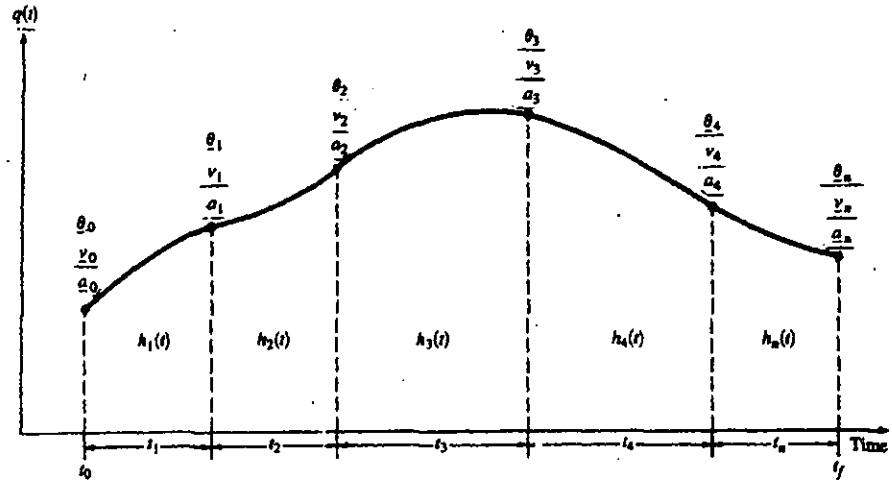

Figure 2.5: 5-cubics spline trajectory [3].

They increase the total number of constraints, which means that all the segments are cubic spline functions, the lowest degree polynomial function that allows continuity in velocity and acceleration. This approach reduces the effort of computation and the possibility of numerical instability.

Therefore, in using five-cubic polynomials an additional knot point is required to reduce each of the end-point fourth order polynomials to a cubic, generating a total of five trajectory segments and six knot points (including initial and final positions) as shown in Figure 2.5. These two extra knot points can be selected between the lift-off and set-down positions. As in (2.11), it is not necessary to know these locations exactly; it is only required that the time intervals be known and that continuity of velocity and acceleration be satisfied at these two locations. Thus, the boundary conditions that this set of joint trajectory segment polynomials must satisfy are:

$$
\left\{
\begin{array}{l}
\text{First } 3rd \text{ degree polynomial} \\
\quad \text{constraints} = \left\{
\begin{array}{l}
\text{Initial position, } q(t_i) \\
\text{Initial velocity, } \dot{q}(t_i) \text{ (normally zero)} \\
\text{Initial acceleration, } \ddot{q}(t_i) \text{ (normally zero)} \\
\text{First knot point position , } q(t_1)
\end{array}
\right. \\
\text{Second, third and fourth } 3rd \text{ degree polynomial} \\
\quad \text{constraints} = \left\{
\begin{array}{l}
\text{Continuity in velocity at } t_j,\ \dot{q}(t_j^-) = \dot{q}(t_j^+) \\
\text{Continuity in acceleration at } t_j,\ \ddot{q}(t_j^-) = \ddot{q}(t_j^+) \\
\text{Continuity in velocity at } t_{j+1},\ \dot{q}(t_{j+1}^-) = \dot{q}(t_{j+1}^+) \\
\text{Continuity in acceleration at } t_{j+1},\ \ddot{q}(t_{j+1}^-) = \ddot{q}(t_{j+1}^+)
\end{array}
\right. \\
\text{Last } 3rd \text{ degree polynomial} \\
\quad \text{constraints} = \left\{
\begin{array}{l}
\text{Last knot point position , } q(t_2) \\
\text{Final position, } q(t_f) \\
\text{Final velocity, } \dot{q}(t_f) \text{ (normally zero)} \\
\text{Final acceleration, } \ddot{q}(t_f) \text{ (normally zero)}
\end{array}
\right.
\end{array}
\right. \tag{2.12}
$$

where $t_j = t_1 ... t_4$ represent the time to arrive at the first, second, third and fourth knots respectively.

The placement of the two additional knot points has received some attention in the past. Some authors [9] have claimed that "the best placement of the new knot points is so close to one end of a segment that nothing happens between the knot and the end-point", but a precise analysis of the problem has yet to be given. Some recent alternatives are reviewed in Chapter 4.

For further details of the calculation of the coefficients for the 4-3-4, 3-5-3, and 5-cubics spline trajectories the reader may refer to [3, 10].

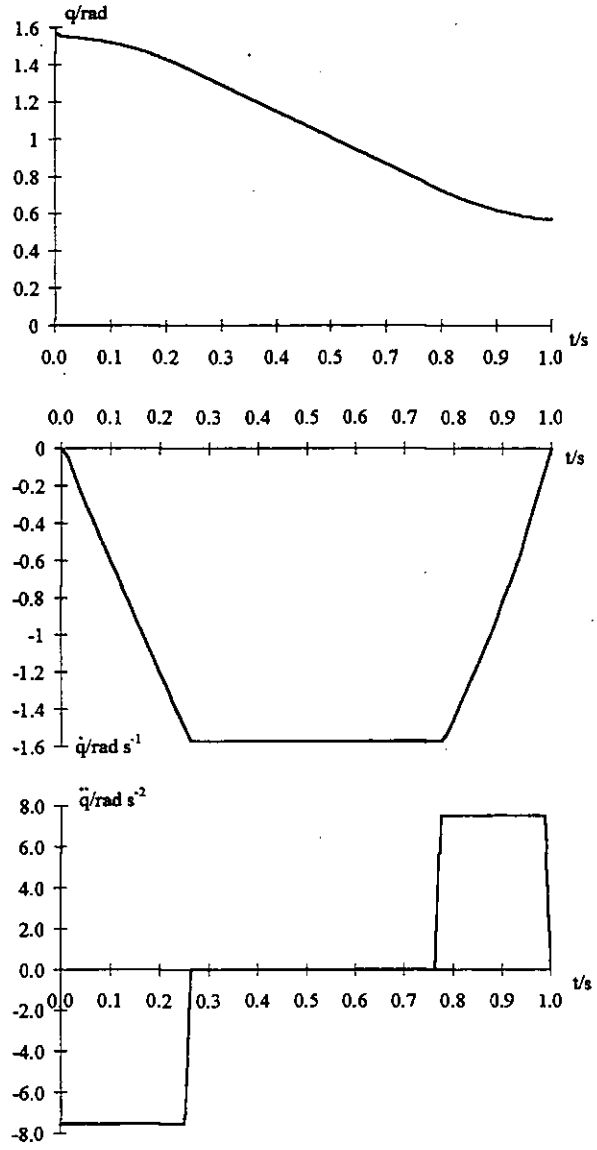

Figure 2.6: Position (top), velocity (middle) and acceleration for bang-coast-bang parabolic spline trajectory.

2.7.4 Linear polynomial with parabolic blends spline

Another choice of polynomial spline is that in which the trajectory scheme is based on a linear interpolation as described by Equation (2.2) in Section 2.3. However, to overcome the drawback of discontinuous velocity and acceleration at the trajectory definition points, straightforward linear interpolation is smoothed by adding a parabolic blend region at each knot point [4].

During the blend portion of the trajectory, constant acceleration is used to change the velocity smoothly, which then remains constant during the linear portion of the motion. The linear function (first order polynomial) and the two parabolic functions (second order polynomials) are splined together so that the entire path is continuous in position and velocity (see "spline" in Section 2.7).

In order to construct the trajectory, the maximum accelerations $\pm\ddot{q}_{max}$ that the joint can generate are assumed to be independent of the arm configuration [12]. As has already been stated, this is a conservative estimate since the maximum acceleration of the joint depends, in general, on the configuration, and hence the dynamics of the arm. This assumption leads to trajectories that are approximately bang-coast-bang, namely the acceleration is either $\pm\ddot{q}_{max}$ or zero (see Equation (2.9) for bang-bang trajectories).

$$
\begin{array}{ccc}
\underline{\text{First Parabolic segment}} & \underline{\text{Mid Linear segment}} & \underline{\text{Last Parabolic segment}} \\
t_i \leq t \leq t_1 & t_1 \leq t \leq t_2 & t_2 \leq t \leq t_f \\
q(t) = q(t_i) + \dot{q}(t_i)t + (\ddot{q}_{max}/2)t^2 & q(t) = q(t_1) + \dot{q}_{max}t & q(t) = q(t_2) + \dot{q}_{max}t - (\ddot{q}_{max}/2)t^2 \\
\dot{q}(t) = \ddot{q}_{max}t & \dot{q}(t) = \dot{q}_{max} & \dot{q}(t) = \dot{q}_{max} - \ddot{q}_{max}t \\
\ddot{q}(t) = \ddot{q}_{max} & \ddot{q}(t) = 0 & \ddot{q}(t) = -\ddot{q}_{max}
\end{array} \tag{2.13}
$$

During the parabolic segments, both of the same duration, the acceleration is kept constant $\pm\ddot{q}_{max}$, the velocity is a linear function of time and the position follows a quadratic polynomial, whereas position varies linearly with time in the middle segment.

Equation (2.13) above shows one such trajectory which further assumes that the joint reaches maximum velocity during the acceleration phase, remaining at that speed during the linear segment (thus describing a trapezoidal velocity profile). Figure 2.6 shows the position, velocity and acceleration profiles for this motion. The switching knot points $q(t_1)$ and $q(t_2)$ can be easily obtained from all these constraints, and in this particular case correspond to those that can make near-full use of the capabilities of the joint actuator. However, slower speed constraints or more relaxed time constraints governing the overall duration of the motion would lead to different switching intermediate knot points.

It is worth noting that in a multibody manipulator with a number of DoF, the execution of a point-to-point parabolic spline trajectory, such as the one just described, is normally executed in a coordinated manner. Hence, the motion of the joints is synchronized so that they all start and stop simultaneously. This means that only the joint that takes longer to perform the motion will run at the maximum speed as described by trajectory (2.13). The rest of the joints will accordingly have their own timings and velocity constraints.

2.8 Summary and Discussion

The trajectory planning strategies discussed so far have a number of points in common:

- They are meant to be efficient with a fast computation time, thus generating the set points that the manipulator must follow in real-time.
- Time considerations are normally either provided by the user or based on an estimated maximum velocity and acceleration of the joints.
- There is no attempt to maximise/minimise any parameter (namely time) along the trajectory.
- They require the unique determination of the parameters of a function which satisfies some given boundary conditions.
- The boundary constraints are entirely based on kinematic considerations, i.e., they must fall within a default maximum velocity/acceleration.

These points raise the following issues for consideration:

- Firstly, the amount of acceleration that the manipulator is capable of at any given time is a function of the dynamics of the arm and the actuator limits and, accordingly, vary across the workspace [6]. Furthermore, most actuators are not characterised by a fixed maximum torque, but rather by a torque-speed curve [4] and a continuous stall torque. Therefore, in order not to exceed the actual capabilities of the device, the boundary conditions in the trajectories planned by the methods described must be chosen conservatively, possibly forcing the robot to be underutilised by not making full use of the speed of the manipulator [13, 14]. Otherwise, large tracking error may result in the servo control of the manipulator.
- Secondly, the parameterised constraint satisfaction approach to trajectory planning presented here has the advantage that it works from simple descriptions. However, modern optimal control theory [15, 16], examined in the next Chapter along with other control strategies, provides a more general approach to constraint satisfaction as will be seen, even when the number of constraints and parameters is different.

These issues, which reflect the attention that has been given in recent work to dynamic considerations and actuator limitations in search of superior trajectory generation methods, are also addressed in this thesis.

References

[1] Luh J.Y.S. and Lin C.S. Optimum path planning for mechanical manipulators. *Journal of Dynamic Systems, Measurement, and Control. Transactions of the ASME*, **103**:142–151, June (1981).

[2] Lin C.S., Chang P.R., and Luh J.Y.S. Formulation and optimization of cubic polynomial joint trajectories for industrial robots. *IEEE Transactions on Automatic Control*, **AC-28**(12):1066–1073, December (1983).

[3] Fu K.S., Gonzalez R.C., and Lee C.S.G. *Robotics: control, sensing, vision, and intelligence.* McGraw-Hill Book Co., (1987).

[4] Craig J.J. *Introduction to Robotics: mechanics and control.* Addison-Wesley Publishing Company Inc., second edition, (1989).

[5] Brady M., Hollerbach J.M., Johnson T.L., Lozano-Perez T., and Mason M.T. *Robot Motion: planning and control.* MIT Press, (1982).

[6] Rivin E.I. *Mechanical Design of Robots.* McGraw-Hill Book Co., (1988).

[7] Paul R.P. The mathematics of computer control manipulator. In *Proceedings of the Joint Automatic Control Conference*, volume **1**, pages 124–131, (1977).

[8] Mutjaba M.S. Discussion of trajectory calculation methods. In Binford T.O. et. al., editor, *Exploratory Study of Computer Integrated Assembly Systems.* Artificial Intelligence Laboratory, Stanford University, AIM 285.4, (1977).

[9] Finkel R.A. Constructing and debugging manipulator programs. Technical Report AIM-284, Artificial Intelligence Laboratory, Stanford University, August (1976).

[10] Chen Y.C. Solving robot trajectory planning problems with uniform cubic b-splines. *Optimal Control Applications & Methods*, **2**:247–262, (1991).

[11] Paul R.P. Modelling, trajectory calculation, and servoing of a computer controlled arm. Technical Report AIM-177, Artificial Intelligence Laboratory, Stanford University, November (1972).

[12] Waters R.C. Mechanical arm control. Technical Report AIM-549, Artificial Intelligence Laboratory, Massachussetts Institute of Technology (MIT), October (1979).

[13] Kim B.K. and Shin K.G. Suboptimal control of industrial manipulators with a weighted minimum time-fuel criterion. *IEEE Transactions on Automatic Control*, **AC-30**(1):1–10, January (1985).

[14] Shin K.G. and McKay N.D. Minimum-time control of robotic manipulators with geometric path constraints. *IEEE Transactions on Automatic Control*, **AC-30**(6):531–541, June (1985).

[15] Bryson Jr.A.E.. and Ho Y.C. *Applied Optimal Control: optimization, estimation, and control.* Blaisdell Publishing Company, (1969).

[16] McCausland I. *Introduction to Optimal Control.* John Wiley & Sons Inc., (1969).

Chapter 3

Fundamentals of Robot Control Strategies

3.1 Introduction

In the preceding Chapter several approaches to the design of manipulator trajectories have been presented. In this Chapter, issuing the commands to the joint actuators that will cause the manipulator to track the specified nominal trajectory is considered. Development of control algorithms for robot manipulators is very much an active area of research and numerous techniques have been explored. The material herein is not intended to be an exhaustive review of current robot control methods because many of the algorithms proposed in the literature are still under active development and/or rely on techniques beyond the scope of this dissertation. Instead, an overview of robot control theory, with a bias toward industrial practice and well established manipulator control algorithms is presented. More specialised applied strategies will be surveyed in the next Chapter, when results taken from the literature are presented. Given the nature of the topic, it is assumed that the reader is familiar with elementary differential equations and the basics of linear control systems, including the Laplace transform and transfer functions, so they will not be listed in full here. Most introductory linear control texts should be sufficient; see (for example) K. Ogata [1] or R. Dorf [2].

The works reviewed in this Chapter are concerned with the design and analysis of control systems for regulation of manipulator end-point position or for tracking of a pre-planned trajectory. It is also possible to design controllers that operate when the (end-effector) tool is in contact with the environment, these are referred to by terms such as compliance, force or impedance controllers. A review of these methodologies can be found in [3, 4].

The fundamental properties of the two alternative approaches to controlling a robotic arm, i.e., closed-loop and open-loop, are analysed in Section 3.2. The conditions under which one is advantageous against the other are examined as a basis for later developments, since both are the object of study in this work. As the theory developed in this thesis relies heavily on state-space control techniques, some of the basic notation associated with modern control system theory is contrasted in Section 3.3 with the conventional control framework.

In Section 3.4 the simplest approach to feedback control, single-axis linear PID control, common in most current commercial robots is considered. The deficiencies of this scheme with regards to compensating for the non-linear manipulator dynamics gives rise to a number of model-based or dynamic control strategies, some of which are discussed in Sections 3.5–3.8. Feedforward control, i.e., a feedback controller supplemented by feedforward information about the manipulator, is discussed in Section 3.5. It is noted that in the specific configuration where the linear feedback control law sends its output through the dynamic model, the exact knowledge of the manipulator parameters leads to a complete cancellation of the non-linear dynamics. However, some degree of adaptive capacity is sought in the frequent case when it is known that manipulator or environment will be subject to variations or the knowledge of the manipulator is not complete. Although various degrees of adaption fall within the rather ill-defined area of Adaptive control, some general techniques are examined in Section 3.6, whereas an alternative robust control approach particularly suited to robotic arms, Variable Structure control, is presented in Section 3.7. In Section 3.8 the type of modern control known as Optimal control is

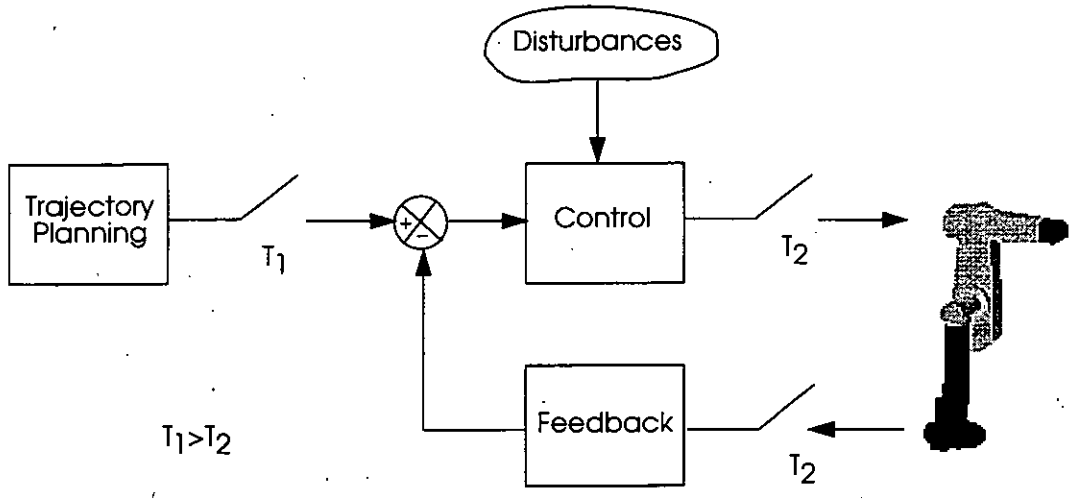

Figure 3.1: Manipulator control block diagram.

analysed. The generic form of this discipline is of extreme importance throughout the analysis methods employed for the remainder of this dissertation. For this reason the framework is quite detailed, and care has been taken in the presentation of the theory underlying the Maximum Principle in Section 3.8.1. This strategy, when applied to the point-to-point manipulator problem forms the class of generally non-linear TPBV problems which is the focus of study in this dissertation.

The analysis of the control problem in Cartesian space is introduced for completeness in Section 3.9, although it is not pursued further. Finally, in Section 3.10, some brief concluding comments are presented.

3.2 General Considerations on Robot Control

In planning the actuator torque commands that will produce the desired motion, the use of information about whether and/or how a particular joint is actually moving will dictate the class of control to be used. Those cases in which the axis output has no effect on the control action are said to be operating in an open-loop manner, whereas the term closed-loop or feedback control applies in the opposite case. The basic feedback control block diagram for a robot manipulator is depicted in Figure 3.1, where T_1 corresponds to the time between trajectory set point updates in the outer loop, and T_2 represents the faster control update inner loop. Typically, the feedback information is compared against the desired position (and maybe velocity also) to compute an error signal. The control system will then calculate the drive torque action to the actuators as a function of the error. This is usually a linear function, as described by the independent joint control in Section 3.4, but need not be, and some other alternatives will be introduced in this Chapter. Feedback control is useful and necessary to compensate for unpredicted disturbances. In particular, when linear feedback control is used alone, the rigid body dynamics of the manipulator are considered as perturbances. These dynamics will in general cause substantial trajectory errors for faster motion, unless gains in the feedback control are made correspondingly higher in an adaptive fashion as described in Section 3.6. Yet there are practical limits as to how high gains can be set, given actuator saturation and stability problems. These problems will be discussed in more detail in Section 3.5.

With regards to open-loop schemes, the common approach is to use a dynamic model of the manipulator to predict actuator torque commands corresponding to the desired joint motion. Such a control technique is termed feedforward control. If the model was a complete and accurate representation of the manipulator and no "noise" or other disturbances were present, continuous use of the dynamic equations of motion of the arm along the desired trajectory would realise the motion. In practice, unfortunately, the manipulator dynamics are not known exactly and there is an inevitable presence of unexpected disturbances which make feedforward control imperfect. Hence, the sole use of such a scheme is not practical for use in real manipulator applications and a feedback controller is also included to compensate for unpredicted disturbances and modelling errors. The two common alternatives of this form of model-based control with feedback compensation will be examined later in the Chapter:

the "feedforward controller" and the "computed torque control" described in Sections 3.5.1 and 3.5.2 respectively, which differ in how the dynamic model is used in conjunction with a feedback loop.

It is important to understand that the term "closed-loop", as applied to manipulators, does not mean that the loop is closed around the master computer (the trajectory planning computer in Figure 3.1). In reality current control practice requires that information about the axis is fed back only to the corresponding joint processors performing the actual higher-bandwidth inner control loop [5, 6, 7]. The master is informed only when the move is completed or if an emergency situation arises (e.g., an obstacle). This control hierarchy means that current manipulators plan trajectories with very little information fed back from the robot and environment, thus severely limiting the capabilities of the trajectory planner to yield improved or optimal trajectories in real-time. This, a direct consequence of lack of processing power in the past, should be expected to change in future generations of robots as more powerful microprocessors are becoming available at reduced cost. Likewise, it is also reasonable to expect that more modern control techniques (some of which are reviewed here) will become increasingly feasible for this same reason. These two remarks will feature strongly in the remainder of this dissertation.

3.3 Modern and Classical Control System Analysis. Preliminary Definitions

The control systems engineering field is a relatively new technological area, with little organised theory existing prior to 1940 and virtually none at all only ten years before that. During the decade of the 1940s, frequency-domain methods made it possible for engineers to design linear closed-loop control systems that satisfied performance requirements. The Laplace transform was utilised to convert the linear differential equations representing the system into an algebraic equation expressed in terms of a complex variable $s = \sigma + j\omega$. By use of the Laplace transform, the transfer function representation of the input-output relationship could be derived. The solution to the differential equation model could then be obtained from the transfer function in s by means of the inverse Laplace transformation, which is made comparatively easy by the use of look-up tables. Moreover, when the differential equation is solved in this fashion, both the transient component and the steady-state component of the solution can be obtained simultaneously. From the end of the 1940s to early 1950s, the root-locus method (due to Evans) and other s-plane methods were fully developed, thus allowing the use of graphical techniques for predicting the system performance without actually solving system differential equations. Furthermore, the alternative frequency response approach which studies the steady-state representation of the system in terms of the real frequency ω was also developed, and several useful techniques for analysis were studied (e.g., Nyquist and Bode).

The frequency-domain approach, in terms of the complex variable s (root-locus methods) or the real frequency variable ω (frequency-reponse methods) constitute the core of classical or conventional control theory. Such an approach generally yields satisfactory (but not optimal) results for single-input single-output (SISO) control systems. However, the limitations of the frequency-domain techniques to meet increasingly stringent requirements on the performance of control systems, the increase in system complexity, and easy access to large-scale digital computers has shifted the emphasis in control design problems to modern state variable control theory since around 1960. It is nevertheless worthwhile keeping in mind that old general-purpose analogue computers provided a convenient method to solve high-order nonlinear differential equations by rewriting the system dynamics into a set of ordinary differential equations, readily implementable and solvable by the old analogue form of computation [8]. While these early differential analysers were superseded by digital computers, the use of analogue computers to solve algebraic equations was, implicitly, an early form of state variable formulation.

3.3.1 State-variable control theory

Time-domain formulation, analysis and synthesis using state variables constitute the foundations of modern control theory. Moreover, modern control is contrasted with classical control in that the former is applicable to complex multiple-input multiple-output (MIMO) systems, which may be linear or non-linear, time-invariant or time-varying, while the latter is applicable only to linear time-invariant SISO systems. The model equation representation in modern control systems is a differential equation,

but written as a set of n first-order coupled state differential equations, usually in vector form. The advantage of this representation is that, in addition to the input-output characteristics, the internal behaviour of the system (state) is also represented. Some additional advantages of this formulation are as follows:

- Computer-aided analysis and design of state models are performed more easily on digital computers for higher-order systems, while the transfer function tends to fail because of numerical problems.
- In state-variable design procedures more information (internal state variables) about the manipulator are fed back; hence a more complete control of the system is possible than with the transfer function approach.
- The time-domain representation of control systems is an essential basis in the field of optimal control, as well as the adaptive and learning control of complex systems. Now that digital computers are becoming cheaper and more compact, they are being used as integral parts of these control systems.
- Even if some state-variable designs are not practical or feasible from the implementation point of view, they still provide a "best" system response which can then be approached using classical design procedures.
- State-variable models are generally required for simulation, that is, digital computer solutions (approximations) of differential equations.

Some state-related definitions used throughout the remainder of this thesis will now be given.

Definition 3.1 *The "state" of a dynamic system is the smallest set of variables (called state variables) such that the knowledge of these variables at $t = t_0$ together with the knowledge of the inputs for $t \geq t_0$ completely determines the behaviour of the system for any time $t \geq t_0$.*

Definition 3.2 *If n state variables are needed to completely describe the behaviour of a given system, then these n variables can be considered the n components of a vector $\boldsymbol{x}$. A "state vector" is thus a vector that determines uniquely the system state $\boldsymbol{x}(t)$ for any time $t \geq t_0$, once the states at $t = t_0$ are given and the inputs $\boldsymbol{u}(t)$ for $t \geq t_0$ are specified.*

Definition 3.3 *This smallest number of variables n required to define the state vector is the "order" of the system, and then a conceptual n^{th} order state space exists, in which the state trajectories are traced by the state variables with increase in time. The simplest multivariable representation is for the two dimensional state and this state-space is referred to as the "phase-plane".*

Remark 3.1 *Although the number of independent state variables (order) required to adequately describe a system will be fixed, the choice of the state variables is not unique. It is convenient to choose state variables that can be directly observed and measured, and maybe associated with energy storage system components, although this is not always possible.*

As indicated above, the general system behaviour representation in state-variable control is a high order (n) differential equation, which may be then reduced to a set of n first-order differential equation by the selection of suitable state variables. These may contain non-linear equations but they may still be written in vector-matrix form. The time derivative of each state variable is expressed as a function of all state variables and system inputs. Hence, for the state variables x_i, $i = 1,...n$ and inputs u_i, $i = 1,...m$

$$\dot{x}_i = f_i(x_1, ..., x_n; u_1, ..., u_m; t) \tag{3.1}$$

or for all the state variables the function vector $\boldsymbol{f}$ is used so that

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) \tag{3.2}$$

The actual output of the manipulator, y_i, $i = 1,...p$ $(p \leq n)$, although dependent on the state vector, need not be identical to the chosen state of the manipulator. Thus, a relationship is also required between the state vector and the output vector

$$y_i = g_i(x_1, ..., x_n; u_1, ..., u_m; t) \tag{3.3}$$

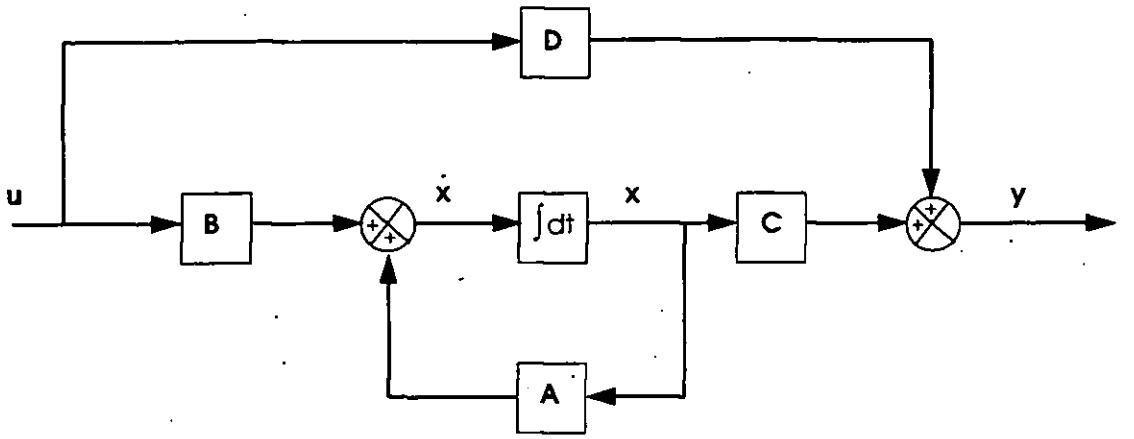

Figure 3.2: Block diagram of LTI control system in state-space representation.

which in function vector form is represented as

$$y = g(x, u, t) \tag{3.4}$$

If vector functions f and g do not involve time t explicitly, then the system is time-invariant, i.e., the system parameters do not vary with time. In that case Equations (3.2) and (3.4) can be simplified to

$$\begin{aligned} \dot{x} &= f(x, u) \\ y &= g(x, u) \end{aligned} \tag{3.5}$$

If f and g in Equation (3.5) are linear functions of x and u then the following state and output equations (state equations in general) of a linear time-invariant (LTI) continuous system arise

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{3.6}$$

where $A \in \Re^n \times \Re^n$ is called the state or system matrix, $B \in \Re^n \times \Re^m$ is the input matrix, $C \in \Re^p \times \Re^n$ is the output matrix and $D \in \Re^p \times \Re^m$ represents direct coupling between input and output and is called the direct transmission matrix. A block diagram representation of Equation (3.6) is shown in Figure 3.2.

3.4 Independent Joint PID Servomechanism

Practically all industrial manipulators currently in use treat each joint of the robot arm as a simple servomechanism [1] based on classical linear feedback control theory [9, 10]. The proportional-plus-integral-plus-derivative compensator (PID) is probably the most commonly used compensator in feedback control systems [11]. The basic structure of this controller for one manipulator joint is shown in Figure 3.3. With $e(t)$ the compensator input (or error) and $\tau(t)$ the output torque (or control action), the PID compensator is defined by the equation

$$\tau(t) = K_p e(t) + K_i \int_0^T e(t)dt + K_d \frac{de(t)}{dt} \tag{3.7}$$

In this control scheme the feedback gains K_p, K_i and K_d are constant and prespecified, usually tuned to perform as a critically damped joint system at a predetermined speed. The "P" control is a pure gain (no dynamics) and gives the controller output a component that is a function of the present state of the system. The "I" control is used to reduce the steady-state error of the system. Since the integrator output depends upon the input for all previous times, that component of the compensator output is

[1] Although this term was originally applied to a system that controlled a mechanical position or motion, it is now often used to describe a control system in which a physical variable is required to track some desired time function. This is in contrast to a regulator control system, where the physical variable is to be maintained at some constant value in the presence of disturbances.

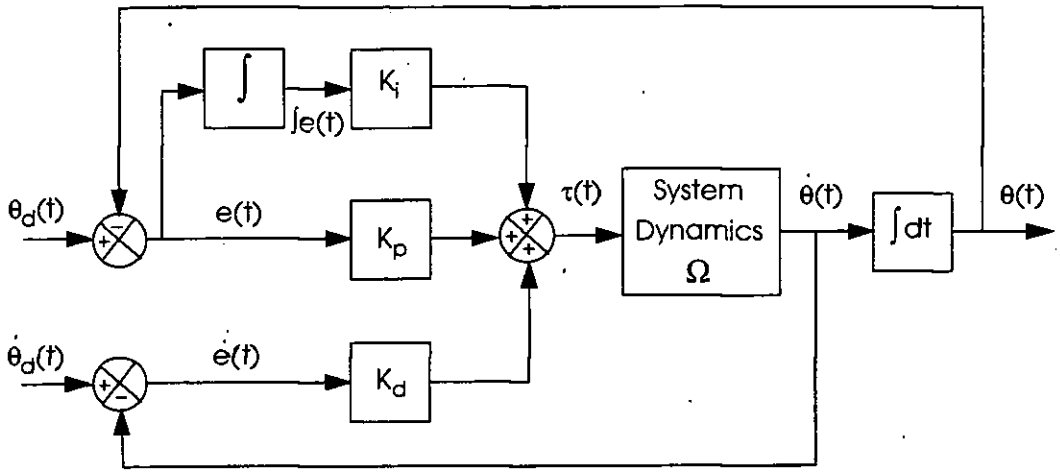

Figure 3.3: Independent-joint PID control.

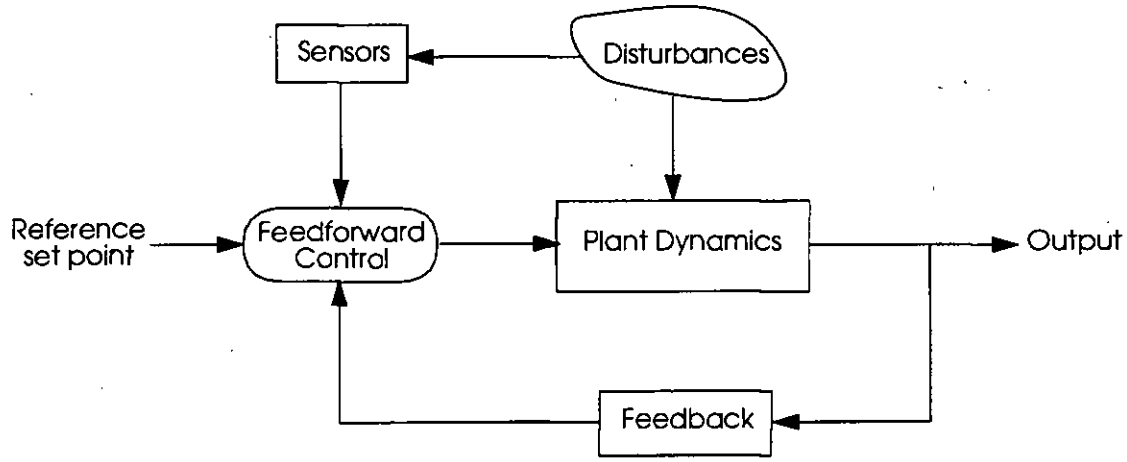

Figure 3.4: Feedforward control.

determined by the past state of the system. The "D" action is a type of phase-lead control and improves the system transient response. The output of the differentiator is a function of the slope of its input and thus can be considered to be a predictor of the future state of the system. Hence the derivative part of the compensation can speed up the system response by anticipating the future. Overall, this type of controller can then be viewed as yielding a control action that is a function of the past, the present, and the predicted future, being employed in control systems in which improvements in both the transient and the steady-state response are required.

3.5 Feedforward Control

It has already been pointed out at the opening discussion of the Chapter that one of the deficiencies of single-axis PID control is that it does not account for the effect of robot dynamics. Consequently, these must be compensated as if they were disturbances. This sets severe limitations in the design of stable feedback controllers. Many proofs of stability for various robot feedback controllers amount to infinite actuator arguments, since it is presumed that actuators do not limit the ability to increase gains to the point where disturbances can be overcome and errors reduced to a desired level. Although this greatly simplifies the controller analysis and design stages by assuming linearity throughout the system, in reality, actuator saturation prevents this easy solution. Moreover, gains cannot be increased to high levels to reduce errors because of potential instabilities that may arise from modelling error, parameter variation, and measurement or command noise [12].

Feedforward control is an approach adopted to reduce the errors that need to be corrected by feedback control. In general control terms, feedforward control is a useful method of cancelling the undesirable effects of disturbances on the manipulator output provided they can be measured by a sensor. By

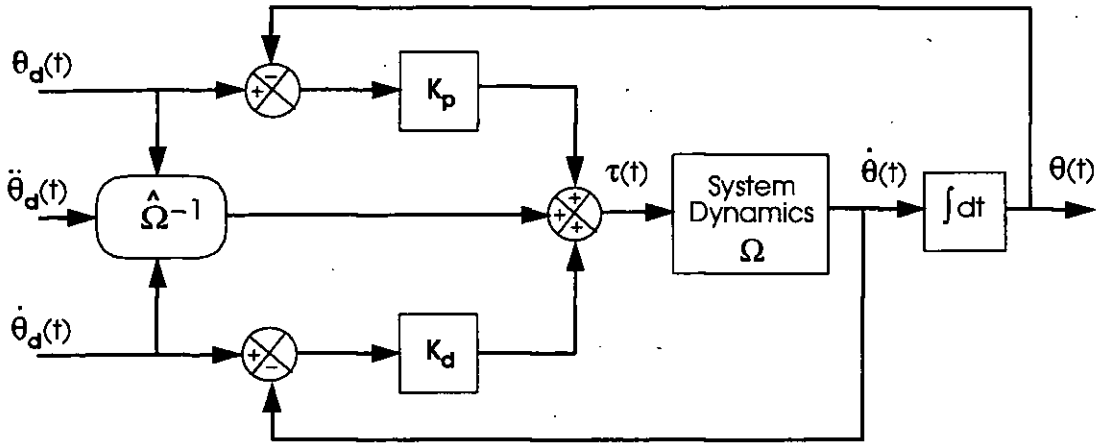

Figure 3.5: Feedforward controller.

approximately compensating for the anticipated disturbances, the corrective action starts before the output is affected, a clear advantage over the usual feedback control where the control action starts only after the output has been affected. Feedforward control can minimise the transient error, but since it is an open-loop strategy, there are limitations to its functional accuracy as outlined before. Feedforward control will neither cancel the effects of unmeasurable disturbances under normal operating conditions nor compensate for any imperfections in the functioning of the feedforward section (mainly the fidelity of the dynamic model of the manipulator $\hat{\Omega}$ to the real manipulator dynamics Ω). It is, therefore necessary that a feedforward control system include a feedback control loop, as shown in Figure 3.4.

In robot manipulator control this translates to equating disturbances to the manipulator dynamics. Hence, based on the desired position, velocity and acceleration provided by the trajectory planner, the dynamic equations of motion [2] are used to predict the output to the actuators that will "cancel" these robot "dynamic disturbances". Note that the trajectory planner must specify not only the desired position and velocity, but the desired acceleration $\ddot{\theta}_d$ also. Depending on how the independent-joint PD is combined with the feedforward control scheme two particular kinds of feedforward control arise, which are discussed next.

3.5.1 Feedforward controller

The feedforward controller, depicted in Figure 3.5, drives the robot actuators according to the following sum

$$\tau = \hat{\Omega}^{-1}(\theta_d, \dot{\theta}_d, \ddot{\theta}_d) + K_p(\theta_d - \theta) + K_d(\dot{\theta}_d - \dot{\theta}) \tag{3.8}$$

where K_p and K_d are now diagonal matrices. The feedforward computation has normally compensated for the dynamics of the robot fairly well, and only small perturbations or unmodelled dynamics remain for the feedback controller to compensate. Hence, the gains of the PD controller can be kept low to avoid stability problems. An important issue to be noted is that the computation of the dynamics is made on the basis of the planned trajectory, and hence can be done off-line according to the control hierarchy of current manipulators. This may have been in the past an important advantage over the computed torque technique examined next, but it is much less of one today due to increasing computational power of real-time control systems.

A disadvantage of the feedforward controller is that the PD portion acts independently of the dynamics and produces perturbations at neighbouring joints, hence degrading system performance. That is to say, a corrective torque at one joint perturbs the other joints, whereas ideally the corrective torques would take these effects into consideration and decouple joint interactions. It is to this latter problem that computed torque control is addressed.

[2] In fact, it is the inverse dynamic problem that will solve for the actuator torques for a given desired joint configuration, i.e., $\hat{\Omega}^{-1}$.

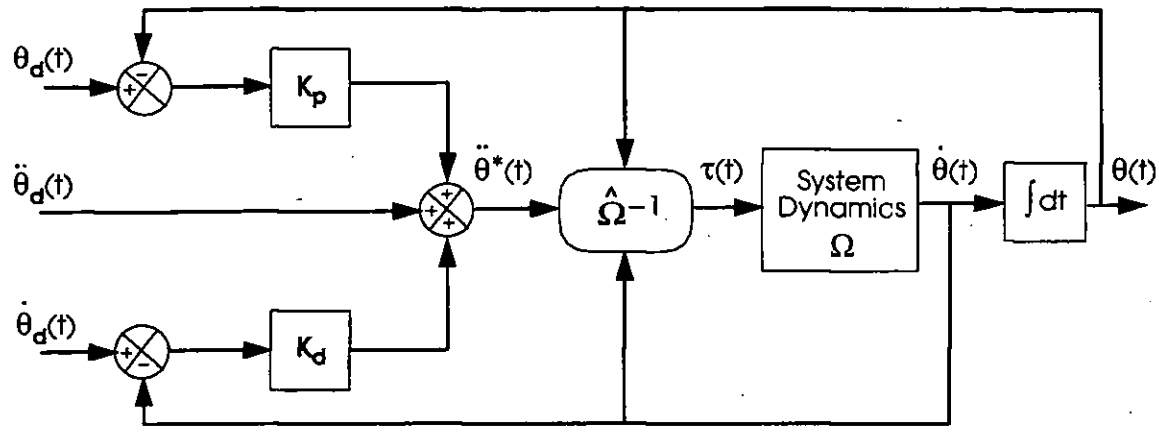

Figure 3.6: Computed torque control.

3.5.2 (Non-linear cancellation) computed torque control

In computed torque control, the linear feedback controller sends its output through the dynamic model as depicted in Figure 3.6. This contrasts with the straight feedback of the linear control portion in the feedforward controller (see Figure 3.5). The feedback control law comprises an independent-joint PD as before, plus the desired acceleration. This yields a corrected nominal acceleration which is then input to the inverse dynamic model, that is.,

$$\begin{aligned} \ddot{\theta}^* &= \ddot{\theta}_d + K_p(\theta_d - \theta) + K_d(\dot{\theta}_d - \dot{\theta}) \\ \tau &= \hat{\Omega}^{-1}(\theta, \dot{\theta}, \ddot{\theta}^*) \end{aligned} \tag{3.9}$$

A close examination of Equation (3.9) will reveal the aforementioned drawback of this control technique with regards to computational requirements. Since the feedforward computation is done on the basis of the actual trajectory, the dynamics computation must be on-line.

It is of interest to observe what happens when the model estimates of the robotic arm parameters are exact, i.e., $\hat{\Omega} = \Omega$. In this case, the non-linear dynamic perturbations are exactly cancelled, leaving a system linear in the position error that can be controlled according to standard linear techniques [3]. Moreover, if the gain matrices K_p and K_d are diagonal, then the closed-loop equations of motion are not only linear, but are also decoupled from one another.

Theorem 3.1 *Let τ, computed according to the computed-torque control law in Equation (3.9), be the control signal of a robotic system with dynamics represented by Ω. Assuming that the dynamic model in Equation (3.9) is exact ($\hat{\Omega} = \Omega$), then the solution of the positional error ($e = \theta_d - \theta$) reduces to a linear second-order system that is independent of the robotic arm parameters.*

It is this feature that makes the Computed Torque control a form of control called `Non-Linear Cancellation` or `Non-Linear Decoupled Feedback Control` [14]. In general form, this non-linear control concept leads, through a suitable partition of the manipulator plant dynamic equations, to explicit non-linear control laws for all subsystems of the plant that correspond to the different variables of motion. The application of these control laws, which are in feedback form and entirely based on the model representation of the manipulator, provide for overall system behaviour in which all outputs of the system are completely decoupled. This, in the case of an industrial robot, yields second-order input-output [4] equations whose characteristic coefficients can be chosen arbitrarily according to linear methods. Unfortunately, non-linear cancellation is not a property inherent to the Feedforward Controller as has already been indicated. Under the same assumption of perfect modelling, the error equation of the system results in a coupled non-linear equation which is a function of both the desired and current state dynamics [13]. Still, some authors consider the Feedforward Controller as a form of non-linear cancellation control strategy [15] because the Feedforward Controller will effectively compensate for a good deal of the manipulator dynamics, thus achieving good trajectory tracking.

[3] Note that this is not the case in the Feedforward Controller, where the dynamic compensation is performed over the desired trajectory, not the current state [13].

[4] The input would be the new input to the overall system, which consists of the physical model of the robot in combination with the feedback non-linear control (similar to the block diagram in Figure 3.6).

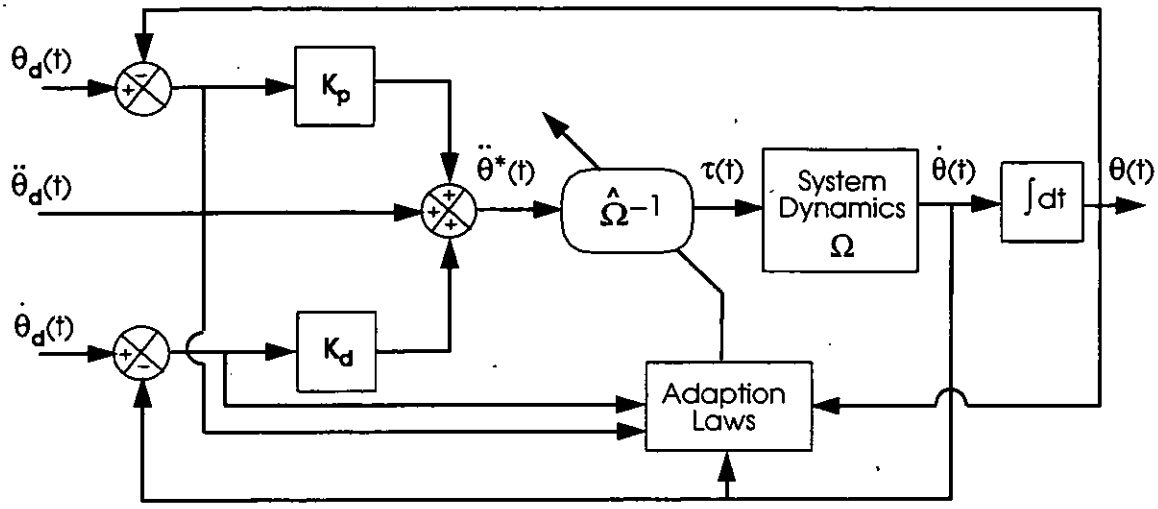

Figure 3.7: Adaptive computed torque control.

3.6 Adaptive Control

In the discussion of model-referenced control, the emphasis has been given to non-linear compensation of the interaction forces between the various joints. Such control systems are established on a knowledge of the manipulator, whose dynamics are presumed accurately modelled. It has been noted, though, that often parameters of the manipulator dynamics are not know exactly and therefore these methods suffer from high sensitivity to errors in the estimates of these parameters. This is particularly true for robot manipulators, where changes in the payload could be significant enough to render these feedback control algorithms ineffective. The result is increasing servo error, reduced servo response and damping, which limits the precision and speed of the end- effector. A number of approaches, some of which are discussed next, have been proposed to develop controllers that are more robust so that their performance is not sensitive to modelling errors. One of the solutions examined is the Adaptive Controller which assesses manipulator and environmental variations and then adapts the control algorithm accordingly so as to maintain a satisfactory response, which is usually judged by some performance index (i.e., maintaining a critical damping over a range of operating velocities and robot configurations). It is important to note that an adaptive controller is a parameter-adjustment loop which is appended to the normal master controller used to control position, velocity and the like. Adaptive control is thus an effort to extend the fixed controller configuration [5] to a time-varying system by adjusting one or more of its parameters (gains, time-constants, etc.).

Probably the most intuitive self-organising scheme corresponds to the block diagram depicted in Figure 3.7. This is essentially an Adaptive Computed Torque controller [16] where the master controller is in itself a model of the real manipulator. This is supervised hy an adaption process which, based on observation of manipulator state and servo errors, readjusts the parameters in the non-linear model until the errors disappear. Hence, such a system would effectively learn its own dynamic properties.

Note, however, that this approach to adaptive model-based control need not be the case for general adaptive controllers. The concept of model-referenced adaptive control (MRAC) is applied not to adaptive systems where the master controller is based on the dynamic equations of the manipulator being controlled, hut to adaption algorithms driven hy the errors between the output of a selected reference model and the actual system outputs [17]. A general control block diagram of the MRAC is shown in Figure 3.8. This form of adaptive control uses a linear time invariant (LTI) second-order differential equation as the reference model for each DoF of the manipulator (where the mass of the payload in taken into consideration by combining it into the final link). The manipulator employs a simple independent joint feedhack controller, whose feedback gains are adjusted in an attempt to make the robot respond like the reference model. As a result, this adaptive scheme requires only moderate computations and an *a priori* accurate model of the system dynamics is not necessary. This fact, however, makes stability considerations of the closed-loop adaptive system difficult, as shown in [17], where the stability analysis was carried out using a linearized model. Furthermore, the adaptability of

[5] Which may or may not be in feedback form.

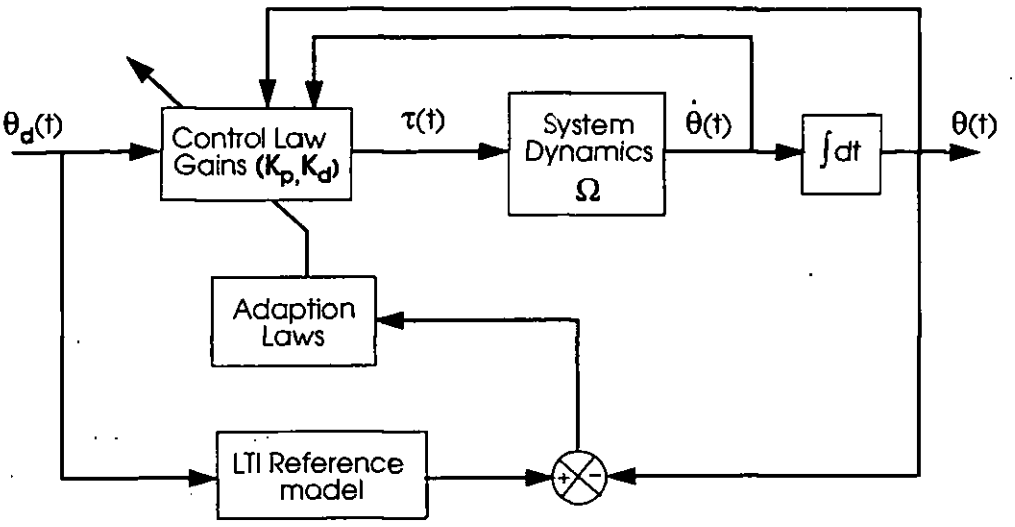

Figure 3.8: Model-referenced adaptive control (MRAC).

the controller can become questionable if the interaction forces among the various joints are severe.

A self-tuning controller for a robotic arm based on an autoregressive model that uses a least-squares criterion to obtain the best fit to the manipulator input-output data has also been proposed [18]. The estimates of the autoregresive model parameters are then used in the design of the control strategy, which assumes that the interaction forces among the joints are negligible.

An alternative adaptive strategy to track a desired trajectory based on the non-linear model of the manipulator dynamics was developed by Lee and Chung [19]. Here, the overall controlled system is characterised by a feedforward component and a feedback component. The former computes the nominal torques from the Newton-Euler (N-E) equations of motion which will supposedly compensate for all the interaction forces between the various joints, hence linearising the control problem along the nominal trajectory. The latter is based on perturbation theory to calculate the perturbation torque which will provide control effort to compensate for small deviations from the nominal trajectory. The design of the feedback law is then based on the linearised perturbation equations, but because it is extremely difficult to obtain analytically the elements of the linearised equation from the N-E equations, a least-squares identification scheme is used to identify these unknown coefficients. With the identification of these parameters, the computation of the perturbation torques is then based on a one-step optimal control law which finds optimal weighting matrices at control rate. As a result, the parameters and feedback gains are updated at each sample period to obtain the combined control effort. A computer simulation of the strategy carried out by the authors to evaluate the performance of this controller with a constant gain PD has compared favourably with the adaptive controller.

3.7 Variable Structure Control

The development of effective adaptive controllers represents an important step towards versatile applications of high-speed and high-precision robots where sensitivity to parameter uncertainties and variations is especially severe. The control of direct-drive robots, for which no gear reduction is available to mask effective inertia variations, is a particularly active area of applied research in adaptive control [20]. However, adaptive methods are based on the assumption that the parameters of the system being controlled do not change too rapidly in comparison with the system time constants. These techniques have proved quite effective when applied, for example, to chemical processes where the process parameters undergo gradual change. In contrast, for robotic manipulators, the system parameters, such as inertia and the effects of gravity, tend to change rapidly as the arm moves from one configuration to another. Although this difficulty has been decreased through advancements in microelectronics, the application of adaptive control methods to robotic manipulators has thus far enjoyed only limited success and is still very much in the research stage [4].

The theory of Variable Structure Systems (VSS) [21] offers a different approach to robust control that appears well suited for the control of robotic manipulators. This is so because VSS are a class of

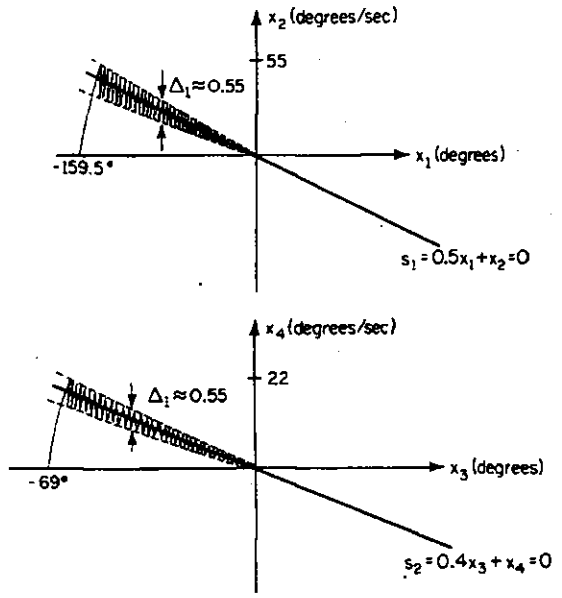

Figure 3.9: Slide mode control state-space trajectories [21].

systems with discontinuous feedback control for which it is not necessary to know the exact robot arm parameters, instead only bounds on these parameters. Hence, Variable Structure Controllers (VSC) are robust in the sense that they are insensitive to errors in the estimates of the parameters as long as reliable bounds on the parameters are known. The salient feature of VSS is the so-called `sliding mode` on the feedback switching surface. A dynamic system is said to exhibit sliding mode when all of its trajectories converge (locally) to a fixed manifold of the state space. The synthesis of the problem, as presented by Young [21], is to choose the discontinuous dynamic feedback controls so that the manipulator system is forced into sliding mode and is also kept there. That is, when in sliding mode, if the system trajectory deviates from the manifold, the control law switches between two discontinuous values, τ^+ and τ^-, of equal magnitude but opposite sign, to return the system to the manifold. This process continues as the solution "zigzags" back and forth across the switching surface, as shown in Figure 3.9 for a two-link manipulator (as described in [21]). It is from the inequalities necessary to guarantee that the state-space trajectories move towards the surface and continue on it after reaching it that the control actions τ^+ and τ^- to be applied are derived [21], and it is due to the inequality nature of these conditions, which are functions of the manipulator dynamics, that only bounds on the manipulator dynamic coefficients are necessary.

To formulate a VSC law, it is helpful to first recast the state vector in terms of the tracking error and its derivative as $\dot{\boldsymbol{x}} = [e, v]$, where $e = \theta - \theta_d$ and $v \equiv \dot{e} = \dot{\theta} - \dot{\theta}_d$. For the regulator problem presented here $\dot{\theta}_d = 0$, thus $v = \dot{\theta}$. A general block diagram of a VSC for a robot manipulator is depicted in Figure 3.10. Even though the natural manipulator dynamics are second order, they can be forced to follow simpler, more well-behaved first order trajectories, s, specified independently for each joint $i = 1, .., n$ in the form

$$s_i = c_i e_i + v_i = 0 \tag{3.10}$$

where c_i is a design parameter called sliding gain. Hence, to slide down this trajectory towards the origin means $e \to 0$ as $t \to \infty$, thus reaching the desired end state. Choosing the appropriate control actions that will keep the manipulator in the switching trajectories implies that, when in sliding mode, the original system is governed by a reduced-order system equation of the form

$$\dot{e}_i = -c_i e_i \tag{3.11}$$

This is called the equation of sliding mode. Equation (3.11) represents n uncoupled first-order linear systems, each one of them representing the dynamics of a single DoF when the manipulator is in sliding mode. Clearly, the overall non-linear interactions in the manipulator dynamics are eliminated completely if the manipulator system is forced into sliding mode. Furthermore, the dynamics of the

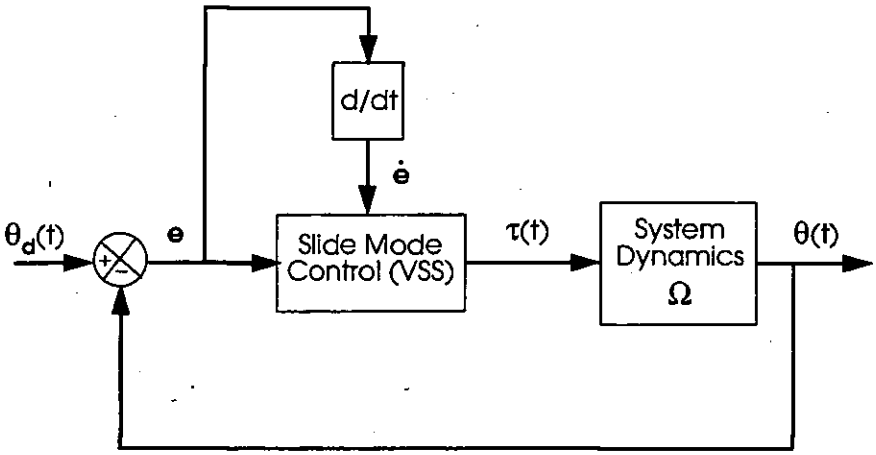

Figure 3.10: Slide mode control.

manipulator in sliding mode depend only on the sliding gain c_i. The manipulator system in sliding mode is therefore insensitive to interactions between the joints and load variations. Recognising that $-c_i$, $i = 1,..,n$ are the eigenvalues of (3.11), the simple choice of positive c_i's in (3.10) guarantees asymptotic stability of the system in sliding mode. Furthermore, the rate at which the error decreases can be controlled through the specification of this design gain.

It should be emphasised that while the the control bound inequalities necessary to guarantee that the trajectories hit the surfaces and remain on them are difficult to analyse, the simulation examples in [21] illustrate that they are not difficult to achieve. In general terms, the inequalities define an (off-line) non-linear programming problem which may be solved by a variety of numerical techniques. It should also be noted that while VSC produces a chattering discontinuous feedback signal that changes sign rapidly (similar to a pulse amplitude modulated signal), simulations carried out in [21] indicate that the joint-position trajectories are observed to be smooth. Hence, although ideally the VSC is switched at an infinite frequency, in reality the manipulator itself acts as a low-pass filter with respect to each variable structure control signal. Other authors have replaced the two discontinuous control actions by some continuous approximation with large slope, hence gradually switching the control signal in a small band around the switching surface, rather than right at the surface [22].

3.8 Optimal Control

The potential advantages of sliding mode control are many, but it is especially the robustness of the method that encourages its adoption to manipulator control problems. However, the VSC method does have its drawbacks. One is that there is no single systematic procedure that is guaranteed to produce a suitable control law. A second is the chattering of the discontinuous control law in sliding mode as discussed above. In addition, it is important to realise that the switching planes are chosen so as to decouple and linearise the dynamics of the manipulator, hence simplifying controller design and improving closed-loop stability. Yet such design methodology does not aim to yield the best phase trajectories along which the manipulator is meant to slide. The control strategy described next accomplishes this task by applying a direct approach especially suited for the synthesis of complex systems called Optimal Control theory.

The idea behind optimal control (or dynamic optimisation, as it is referred to by some authors) is fundamentally different to that pursued by the strategies described in the foregoing portions of this Chapter. In those methods, emphasis is placed upon determining the design parameters of an "acceptable" system that will, customarily, drive the tracking or regulating error to zero. Acceptable performance is generally defined in terms of time and/or frequency domain criteria such as rise time, peak overshoot, gain and phase margin, bandwidth and the like, and various integrals of the error are usually employed as the basis of determining the quality of the control. These performance indices are all similar in that no penalty is included for the cost of the control itself or other parameters like the state or time. Radically different performance criteria must be satisfied, however, by the complex MIMO systems required to meet the demands of modern technology. For example, the design of a

manipulator control system that minimises the expenditure of control energy or the period of time necessary in performing a task with bounded control input are not amenable to solution by classical methods. In that respect, dynamic optimisation can be best described as *a control design methodology concerned with obtaining the best possible response of a process that satisfies the physical constraints imposed and at the same time minimises (or maximises [6]) some scalar index of performonce "IP" which maybe a function of all the state and control variables of the system.*

All standard dynamic optimisation techniques depend in some sense upon classical calculus of variations methods to derive a set of necessary conditions that must be satisfied by an optimum control law. These conditions for optimality lead to the (generally non-linear) classical TPBV problem that must be solved to determine an explicit expression for the optimal control. In fact, until 1957, classical calculus of variations methods provided essentially the only approach to TPBV problems. It was about then when Pontryagin [23, 24], motivated by an interest in problems with bounded control or state variables, postulated the Maximum Principle (MP), and almost simultaneously Bellman [25] suggested Dynamic Programming (DP) based upon the principle of optimality. Although these techniques provide an analytical expression for the optimal control law, their applications are essentially limited to linear time-invariant systems with quadratic form IP's. Even in such cases, analytical solutions are prohibitively complex for high-order systems ($n \geq 3$).

Computational solutions, made available shortly after 1957 by a number of researchers (fundamentally mathematicians), introduced iterative computational procedures based upon small-scale linearisation, and development of numerical procedures has been extremely rapid since 1960 in parallel with the advent of digital computers. These methods are readily applicable to a wide range of boundary-value problems, including cases where the state equations are non-linear and/or time-varying; the IP is analytically intractable (non-quadratic in form or dependent upon time-varying coefficients); or constraints are functionally dependent upon state and time, often non-linearly so. Moreover, even when analysis is relatively straightforward, computational solutions are often less expensive. In these so-called "direct methods" for solving the TPBV problem, the procedure is to generate a sequence of solutions, each superior to those preceding it as measured by the IP, converging toward the desired optimum solution. Some of these techniques include the steepest descent, variation of extremals and quasi-linearisation [26], all of which determine an open-loop optimal control associated with a specified set of initial conditions.

Realistically then, the MP or the DP approach must be viewed as a starting point for obtaining numerical solutions to optimal control problems. From these, knowledge of the form of the optimal control (if it exists) is obtained and a statement of the TPBV problem which, when solved, yields an explicit relationship for the optimal control. Furthermore, if the optimal control law is in feedback form, *all* of the states must be first available for measurement (or estimation). Therefore, it is understandable that optimal control theory does not, at the present time, constitute a generally applicable procedure for the design of simple controllers, even more so if real-time constraints are an issue in the design of the controller, as in the case of robot manipulators. These limitations may preclude implementation of the optimal control law (they certainly have in the past); however, the theory of optimal control is still useful because:

1. Knowing the optimal control law may provide insight helpful in designing a suboptimal, but easily implemented controller.

2. The optimal control law provides a standard for evaluating proposed suboptimal designs, i.e., a quantitative measure of performance degradation caused by using a suboptimal controller can be established.

There are many techniques that could be presented here. In the interest of clarity and the concern of this research work, however, the fundamental concepts of Pontryagin's MP will be presented next, along with a general discussion about Bellman's DP and Kalman's solution for linear systems.

[6] Any optimisation problem which can be represented as a maximisation problem (i.e., maximise A) can equally well be represented by a minimisation problem (minimise $-A$). Throughout this dissertation, any results obtained for a minimisation apply also for a maximisation.

3.8.1 Pontryagin's Maximum Principle

In 1956, the Russian mathematician Pontryagin and his coworkers Boltyanskii and Gamkrelidze, hypothesised the MP as a generalisation of the calculus of variations to study the optimal control of systems in which there is a constraint of some kind on the instantaneous value of the control input [23, 24]. A complete derivation of the MP involves extensive detail and is beyond the scope of this thesis. This section concentrates instead upon development, rather than proof, of a general MP statement, while its practical application to the manipulator control problem under investigation here will be presented later in Chapter 6. For a more rigorous presentation the reader may refer to [26, 27].

The problem is that of finding an optimal control $\boldsymbol{u}(t)$ for a system described by Equation (3.2) which takes the system from an initial state $\boldsymbol{x}(t_0)$ to a specified final state $\boldsymbol{x}(t_f)$ while minimising a general IP of the form

$$J(t) = \int_{t_0}^{t_f} G(\boldsymbol{x}, \boldsymbol{u}, t)dt \tag{3.12}$$

where the IP integrand G is referred to as the loss function, and represents a measure of instantaneous loss change from ideal performance. Therefore, the IP is interpreted as the cumulative loss. The typical problem which can be solved by the MP is one in which the control vector, with components u_i, $i = 1, ...m$, is constrained so as to lie in a closed bounded admissible region $\mathcal{U}$ in the m-dimensional vector space of the control inputs by the relation

$$U_{i_{min}} \leq u_i \leq U_{i_{max}} \tag{3.13}$$

It is possible that the admissible region may vary with time, but it will be considered fixed here for simplicity. This will be later reviewed in Chapter 6 during the actual design of the optimal controller when practical issues are taken into consideration. The artificial state variable x_0 is added to the original problem with n-state variables (and hence becoming $(n+1)$-order) to force the problem into the MP framework. This variable is the performance index [7] itself, i.e., $x_0 = J \mid x_0(0) = 0$, so that $\dot{x}_0 = G(\boldsymbol{x}, \boldsymbol{u}, t)$. The MP requires a set of auxiliary dependent variables or Lagrange multipliers [8], $p_i(t)$, defined by the following linear and homogeneous form

$$\dot{p}_i = -\sum_{j=0}^{n} \frac{\partial f_j}{\partial x_i} p_j \qquad i = 0 \ldots n \qquad \text{or} \qquad \dot{\boldsymbol{p}} = -\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right)^T \boldsymbol{p} \tag{3.14}$$

where $\dfrac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}$ is the same Jacobian matrix discussed later in Section 3.9.2 to show the relationship between manipulator Cartesian and joint velocities.

The extended state equation and the Lagrange multiplier vector are then combined into a new scalar function $\mathcal{H}$, called the Hamiltonian for its analogy to Hamilton's equation of motion of a mechanical system, and defined as

$$\mathcal{H}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}, t) = \boldsymbol{p}^T \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) = \sum_{i=0}^{n} p_i f_i(\boldsymbol{x}, \boldsymbol{u}, t) \ . \tag{3.15}$$

The MP states that:

Definition 3.4 *The control input $\boldsymbol{u}$ which, while remaining in the permissible closed bounded region, minimises J, must maximise at each time instant the Hamiltonian $\mathcal{H}$.*

i.e., the MP sets a general requirement upon $\boldsymbol{u}$ according to [9]

$$\mathcal{H}_{max}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}, t) = \max_{\boldsymbol{u} \in \mathcal{U}} \mathcal{H}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}, t) \tag{3.16}$$

where the subscript $\boldsymbol{u}$ indicates the variable which is varied in order to achieve the maximisation of $\mathcal{H}$ for given $\boldsymbol{x}$, $\boldsymbol{p}$. Hence, rather than providing a direct solution to the optimal control problem, the MP produces the result in terms of the solution of another set of differential equations, that is, Equation (3.14). Whether this system of equations in terms of the auxiliary variables can be solved

[7] Notation (t) for time-variant parameters will be dropped for simplicity since that applies to all variables involved unless otherwise stated or when required for a more comprehensive exposition.

[8] Also commonly referred to as costate, adjoint or auxiliary variables in the literature and hereafter.

[9] This restriction in the control variable would not apply in the general unconstrained problem treated with classical variational methods.

depends upon the existence of initial conditions for the differential equations. The design of the optimal control law requires the control $\boldsymbol{u}$ to be in terms of the state vector $\boldsymbol{x}$ and time t. However, according to definition 3.4, maximising $\mathcal{H}$ with respect to $\boldsymbol{u}$ results in a control law in terms of the auxiliary variables $\boldsymbol{p}$. This coupling between the state and the Lagrange multiplier can be accomplished by using Equation (3.14). Differentiating the Hamiltonian, Equation (3.15) with respect to $\boldsymbol{p}$, we obtain

$$\frac{\partial \mathcal{H}}{\partial p_i} = f_i(\boldsymbol{x}, \boldsymbol{u}, t) \qquad i = 0 \ldots n \tag{3.17}$$

and with respect to $\boldsymbol{x}$, we obtain

$$\frac{\partial \mathcal{H}}{\partial x_i} = \sum_{j=0}^{n} \frac{\partial f_j}{\partial x_i} p_j \qquad i = 0 \ldots n \tag{3.18}$$

Substituting Equation (3.17) into the $(n+1)$-order state Equation (3.1)

$$\dot{x}_i = \frac{\partial \mathcal{H}}{\partial p_i} \qquad i = 0 \ldots n \qquad \text{or} \qquad \dot{\boldsymbol{x}} = \frac{\partial \mathcal{H}}{\partial \boldsymbol{p}} \tag{3.19}$$

And substituting Equation (3.18) into the auxiliary variable Equation (3.14) we have

$$\dot{p}_i = -\frac{\partial \mathcal{H}}{\partial x_i} \qquad i = 0 \ldots n \qquad \text{or} \qquad \dot{\boldsymbol{p}} = -\frac{\partial \mathcal{H}}{\partial \boldsymbol{x}} \tag{3.20}$$

Equations (3.19) and (3.20) are written in the Hamiltonian canonical form and relate the auxiliary variables to the original state variables , hence completely stating the MP in terms of $\mathcal{H}$. In view of this fact, the partial Pontryagin's MP stated in definition 3.4 may be extended as follows

Definition 3.5 *Let $\boldsymbol{u}$ be an admissible control input to a system characterised by Equation (3.2) such that it is desired to transfer the system state from an initial state $\boldsymbol{x}(t_0)$ to a specified final state $\boldsymbol{x}(t_f)$ (where some of these components may not be constrained to fixed values and might be missing). If $\boldsymbol{u}$ is optimal, in that it minimises the value of the variable J defined by Equation (3.12), then there exists a non-zero continuous vector $\boldsymbol{p}$ which optimally satisfies Equations (3.19) and (3.20) for all $t \in [t_0, t_f]$. In addition, $\mathcal{H}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}, t) = \mathcal{H}_{max}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}, t)$* [10].

The answer to this problem requires the solution of $2n+2$ first-order differential equations, (3.19) and (3.20), which in turn require the specification of an equal number of boundary conditions. Since these boundary conditions are split, i.e., some are given for t_0, and some for t_f, such problems are called two-point boundary-value (TPBV) problems, and as stated in the introduction to this section, they are in general rather difficult to solve. Unless the system is of low-order, time-invariant, and linear, there is little hope of solving the TPBV problem analytically [26]. Hence, iterative methods are frequently employed. Typically the initial and final conditions are known for the state variables but are often not known for the Lagrange multipliers. Thus, iterative methods are usually based on generating good guesses of the initial auxiliary variables to converge a solution for the state and costate differential equations (some will be presented later in Section 4.3.3).

Furthermore, Pontryagin has shown that a necessary condition for optimality (obtained from the calculus of variation's Transversality Condition [28], introduced to find all the boundary conditions which are required for the solution of the problem) is that [11]

$$p_0(t_f) = -1 \tag{3.21}$$

From Equation (3.20) we see that the time derivative of p_0 is zero, so that p_0 is constant and Equation (3.21) is satisfied at every instant $t \in [t_0, t_f]$. Hence, the zeroth term of Equation (3.19) and (3.20) is superfluous, and the canonical form of these equations can be restricted to $2n$ expressions, i.e., for $i = 1 \ldots n$. If a solution can be attained, then the full specification of the boundary conditions described by initial and final state suffices to obtain it.

It should be emphasised that the MP conditions described by definition 3.5 constitute a set of *necessary* conditions for optimality; these conditions are not, in general, sufficient. Further to these conditions, Pontryagin also derived other necessary conditions for different problems (e.g., t_f fixed or free, $\boldsymbol{x}(t_f)$ partially or completely specified, etc.). However, since they are not relevant to the work described in this dissertation, they will not be stated here. For a fuller exposition see, for example, [26].

[10] This condition is sometimes expressed as $\mathcal{H}(\boldsymbol{x}^*, \boldsymbol{u}, \boldsymbol{p}^*, t) \leq \mathcal{H}(\boldsymbol{x}^*, \boldsymbol{u}^*, \boldsymbol{p}^*, t)$ where * indicates optimal values.

[11] Because Equation (3.14) is linear and homogeneous, $p_0(t_f)$ can be taken to be any negative value and the other components of $\mathbf{p}$ will be scaled up or down to suit. If the minimum principle was in use instead, $p_0(t_f)$ is taken to be 0 or +1 (or in general any other positive value).

3.8.2 Other optimal control strategies

The basic concepts of DP and the "Principle of Optimality" upon which it is based were introduced by Bellman [25] in 1957. In its most general form, DP is a tool for determining optimum solutions to multi-stage decision problems, i.e., a process where a choice is required between two or more alternatives at discrete intervals in time. The basis for the best decision taken among those available at each stage is again a performance index. In essence DP finds the optimum solution by testing all *acceptable* decision sequences at each stage to determine the optimum one, usually working backward from the final stage to the initial one. Bellman originally stated the principle of optimality in this manner:

Definition 3.6 *An optimal policy has the property that whatever the initial state and initial decision are, the remaining decision must constitute an optimal policy with regard to the state resulting from the first decision.*

By employing the principle of optimality, DP provides an organised approach to such a problem, and obtains a considerable reduction in the number of calculations necessary compared with the complete direct enumeration of all the possibilities. However, even when efficiently organised, this procedure often requires extensive and sometimes prohibitive storage locations (what Bellman calls the *curse of dimensionality*) and computational capability, limiting the problems for which DP provides a practical solution. The derivation of DP also reveals another important concept - the "Imbedding Principle". Bellman placed the emphasis on determining the optimal decision to be made at any stage of the system, rather than at some fixed state. The desired optimal trajectory is a path which is a function of the desired starting point $\boldsymbol{x}(0)$. This imbedding process is accomplished at the $(N-k)^{th}$ stage by not simply determining the optimal path from the state $\boldsymbol{x}(N-k)$, but rather obtaining the optimal path from all possible states of the $(N-k)^{th}$ stage. This means that the optimal policy and minimum costs for a k-stage problem are also contained (or imbedded) in the results for an N-stage process, provided that $N \geq k$.

The principle of optimality can be mathematically formalised in terms of the multi-stage decision process. Assume an N-stage decision process with fixed initial and final conditions and an IP $J_{k,k+1}(\boldsymbol{x}(k), \boldsymbol{u}(k))$ for evaluating decisions, where index k denotes the current stage, $\boldsymbol{x}(k)$ is the current state at stage k and $J_{k,k+1}$ is the cost incurred in moving to the next stage by applying a $\boldsymbol{u}(k)$ control action. The principle of optimality leads to work this problem backwards by considering the last stage, N, first. Optimum decisions are determined for each possible state at the last stage. Next, all possible states and decisions at the $N-1$ stage are tested, the optimum decisions are stored for each possible state, and the procedure is repeated backward to the first decision stage. The recurrence relation of DP becomes

$$\begin{aligned} C^*_{k,N}(\boldsymbol{x}(k), \boldsymbol{u}(k)) &= J_{k,k+1}(\boldsymbol{x}(k), \boldsymbol{u}(k)) + J^*_{k+1,N}(\boldsymbol{x}(k+1)) \\ J^*_{k,N}(\boldsymbol{x}(k)) &= \min_{\boldsymbol{u}(k)} C^*_k(\boldsymbol{x}(k), \boldsymbol{u}(k)) \end{aligned} \tag{3.22}$$

which is ideally suited for digital computer solution but is extremely demanding on computational resources. Repeated application of (3.22) allows development of the optimal policy one stage at a time backward through the N stages. The end result is a family of optimum solutions for all possible starting conditions. That member of the family corresponding to the given initial conditions $\boldsymbol{x}(0)$ is the desired optimum solution. The presence of state and control constraints generally complicates the application of variational techniques; however, in DP, state and control constraints reduce the range of values to be searched and thereby simplify the solution.

Few engineering problems involve n-stage decision problems directly, although many can be constructed as such if desired. For example, a TPBV problem may be treated using DP by quantising time, state and control into a grid and solving this near-equivalent discrete problem. Accuracy of the mathematical model depends upon quantisation increments in those variables. Decreasing grid size rapidly increases the number of combinations to be checked, so a compromise is required. On the other hand, application of the optimality principle to the continuous TPBV problem provides necessary conditions for an optimal solution in the form of a partial differential equation, generally non-linear and time-varying, called the Hamiltonian-Jacobi-Bellman (H-J-B) equation. This is stated, without proof [26, 29], as

$$0 = J^*_t(\boldsymbol{x}(t), t) + \mathcal{H}(\boldsymbol{x}(t), \boldsymbol{u}^*(\boldsymbol{x}(t), J^*_{\boldsymbol{x}}, t), J^*_{\boldsymbol{x}}, t) \tag{3.23}$$

where $J_{\boldsymbol{x}}^* = \frac{\partial J^*}{\partial \boldsymbol{x}}$ and $J_t^* = \frac{\partial J^*}{\partial t}$ and $\mathcal{H}$ is a Hamiltonian defined similarly as in the case of the MP.

Although this equation is directly solvable for linear systems with quadratic-form IP's [12] (i.e., the linear regulator problem), in general (non-linear plant or non-quadratic-form IP's) the H-J-B equation must be solved by numerical techniques similar to those employed in finding an answer to the MP. However, the MP solution was reduced to the solution of two non-linear ordinary differential equations, (3.19) and (3.20), which are easier to solve than the non-linear partial differential equation obtained using DP. Furthermore, it is important to point out that DP applies directly to non-linear systems with arbitrary constraints on control or state, but the H-J-B equation for continuous systems was not derived in any such generality. Of the continuous variational methods, only the MP applies to such systems. Although control can be held within bounds by inclusion of sufficient penalties in the IP, this solution modifies the problem and tends to yield suboptimal performance relative to the original objectives. It is also interesting to point out that the necessary conditions of the MP can, in some cases, be obtained from the DP by a change of variables [30], although this derivation assumes the existence of some derivatives that in many cases do not exist.

The last of the solutions presented is actually an important subclass of optimal control problems - the Linear Quadratic Regulator (LQR) problem. It has already been mentioned that for linear systems with quadratic IP's the solution using DP leads to a differential equation of the *Riccati* type. Kalman [31] reached the same equation by the use of variational methods. Quadratic IP's are used since they lead to convenient mathematical operations in the determination of the optimal controller. Kalman considered a general quadratic IP of the form

$$J = \frac{1}{2}\boldsymbol{x}^T(t_f)\boldsymbol{P}\boldsymbol{x}(t_f) + \frac{1}{2}\int_{t_0}^{t_f}[\boldsymbol{x}^T(t)\boldsymbol{Q}(t)\boldsymbol{x}(t) + \boldsymbol{u}^T(t)\boldsymbol{R}(t)\boldsymbol{u}(t)]dt \tag{3.24}$$

which can be physically interpreted as the desire to maintain the state vector close to the origin without and excessive expenditure of control effort. The matrices $\boldsymbol{P}$, $\boldsymbol{Q}(t)$ and $\boldsymbol{R}(t)$ are symmetric, and the inverse of $\boldsymbol{R}(t)$ must exist. For the linear system of this general form, Kalman asserts that the optimal control function is a linear function of the state vector. In fact, Kalman proposes the negative feedback function

$$\boldsymbol{u}^*(t) = -\boldsymbol{R}^{-1}(t)\boldsymbol{B}^T(t)\boldsymbol{K}(t)\boldsymbol{x}(t) \equiv \boldsymbol{F}(t)\boldsymbol{x}(t) \tag{3.25}$$

Notice that even if the plant is fixed, the combined feedback matrix $\boldsymbol{F}(t)$ is time-varying. In addition, measurements of *all* the state variables must be available to implement the optimal control law (3.25).

The matrix $\boldsymbol{K}(t)$ must satisfy the same matrix *Riccati* differential equation mentioned earlier which has the following form

$$\dot{\boldsymbol{K}}(t) = \boldsymbol{K}(t)\boldsymbol{B}(t)\boldsymbol{R}^{-1}(t)\boldsymbol{B}^T(t)\boldsymbol{K}(t) - \boldsymbol{K}(t)\boldsymbol{A}(t) - \boldsymbol{A}^T(t)\boldsymbol{K}(t) - \boldsymbol{Q}(t) \tag{3.26}$$

where $\boldsymbol{A}(t)$ and $\boldsymbol{B}(t)$ are the matrices describing the linear plant as expressed by Equation (3.6) but in a time-varying fashion, and $\boldsymbol{K}(t_f) = \boldsymbol{P}$. Although the optimal control law for the linear regulator problem has been shown here to be a LTI function of the system states, under certain conditions the control law becomes time-invariant [26]. A generalisation of the linear regulator problem to the linear tracking problem can also be found in [26].

3.9 Cartesian-based Control

It was already discussed in Chapter 2 how the control strategy used in most robots is based on the ability to govern the position of the joints, where the most natural reference frame for a robot is defined by its joints. Hence, planning trajectories in joint-space is an advantageous approach and is the most widely used (also employed in this investigation). However, very often the manipulator end-effector is required to follow straight-lines and other path shapes which are readily described in terms of Cartesian (tool) coordinates without the need to consider the particular geometry of the robot until the joints positions are required. Furthermore, since joint-interpolated strategies are not generally well-defined to follow a Cartesian path, the resultant joint trajectories can only approximate the Cartesian path (see Section 2.2). In the following sections two basic techniques to achieve Cartesian-based control are introduced. Although such approaches are not currently used in industrial robots, mainly due to

[12] In that case, the H-J-B equation leads to a differential equation of the *Riccati* type.

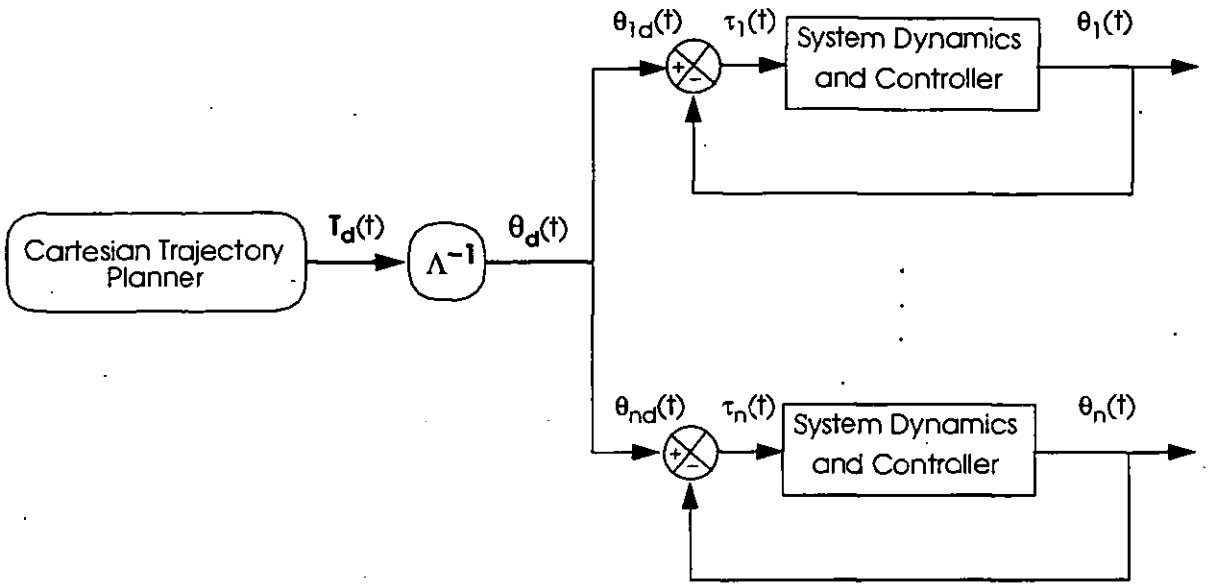

Figure 3.11: Joint-based control with Cartesian trajectory conversion.

analytical complexity and computational demand, they are nevertheless an interesting and active area of research. As the schemes rely heavily on the specification of the Cartesian knot points, a preliminary discussion about the representation of the end-effector in Cartesian space is first given.

3.9.1 Representation of tool configuration

It is generally assumed that the tool configuration is represented by the pair $\{\boldsymbol{r},\boldsymbol{R}\}$, where $\boldsymbol{r} \in \Re^3$ represents the tool position and $\boldsymbol{R} \in \Re^3 \times \Re^3$ represents the tool orientation, both relative to a rectangular coordinate frame at the base of the robot. The pair is usually combined into a $\Re^4 \times \Re^4$ homogeneous matrix $\boldsymbol{T}$ for conceptual representation although, from the computational point of view, this is evidently wasteful of computer memory and they are actually stored separately [3]. Specifying the position of the end-effector with a translation three element vector $\boldsymbol{r}$ is natural and convenient. However, specifying tool orientation with a 3×3 rotation matrix $\boldsymbol{R}$ is, at best, awkward, because two-thirds of the information that must be provided is redundant. An orthogonal set of three unit vectors can be completely specified by three angles such as the Euler angles associated with Cylindrical representation or the Roll-Pitch-Yaw linked to the Spherical coordinate system [7]. Another compact approach of specifying the rotations with only three angles is the Tool-configuration vector [4]. Thus, independent of the method chosen, tool position and orientation can then be specified in a more convenient way as a vector $\boldsymbol{w} \in \Re^6$ where the first three components represent the tool position $\boldsymbol{r}$, while the last three angles represent the tool orientation. Although Cartesian trajectory algorithms usually provide control set points as $\boldsymbol{T}$, it might be more suitable to the controller to transform this representation to the more compact $\boldsymbol{w}$ as described in the following sections.

3.9.2 Joint-based control with Cartesian trajectory conversion

The architecture for this scheme is shown in Figure 3.11. The basic feature of this approach is the conversion of the tool-configuration trajectory $\boldsymbol{T}(t)$ generated by some Cartesian trajectory planner to a corresponding joint-space trajectory $\boldsymbol{\theta}(t) \in \Re^n$ (where n represents the number of DoF of the manipulator) through the non-linear inverse kinematic transformation $\boldsymbol{\Lambda}^{-1}$, that is,

$$\boldsymbol{\theta}(t) = \boldsymbol{\Lambda}^{-1}(\boldsymbol{T}(t)) \tag{3.27}$$

This is then followed by some kind of linear joint-based servo scheme as described in Section 3.4. Although for simplicity, velocity feedback or acceleration reference are not shown in Figure 3.11, they might also be part of the controller. In that case, further kinematic transformations are required to

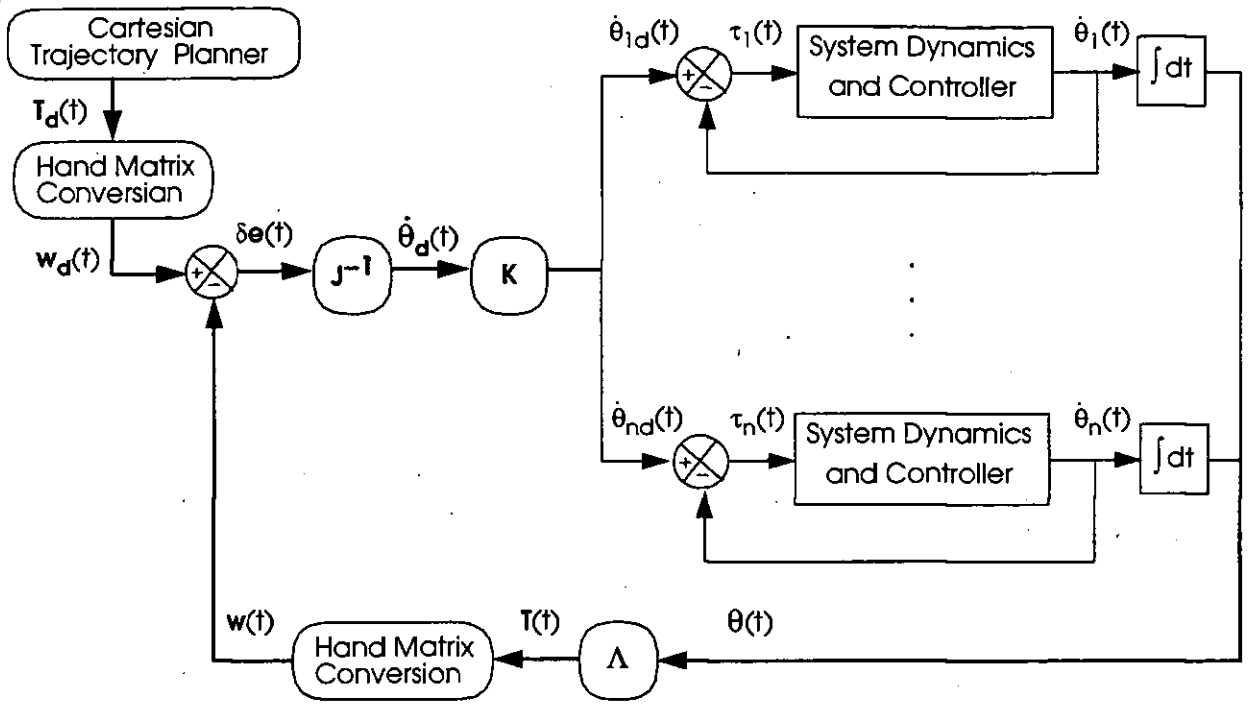

Figure 3.12: Resolved motion rate control (RMRC).

determine the rates $\dot{\theta}(t)$ and $\ddot{\theta}(t)$ at which the individual joints should be driven

$$\begin{aligned} \dot{\theta}(t) &= J^{-1}(\theta(t))\dot{w}(t) \\ \ddot{\theta}(t) &= \dot{J}^{-1}(\theta(t))\dot{w}(t)+J^{-1}(\theta(t))\ddot{w}(t) \end{aligned} \tag{3.28}$$

The matrix J relating the hand (tool) velocities to the joint velocities is called the Jacobian. Essentially, it allows the computation of a differential change in the tool coordinate frame due to a differential change in the position of the manipulator's joints. If the matrix is invertible (it might not be square and/or non-singular) then it is possible to calculate joint velocities and accelerations given the hand velocities and accelerations according to Equation (3.28) above. Note the compact representation $w(t)$ required for this transformation. The trajectory conversion process is quite difficult if it is to be determined analytically. Moreover, for efficiency reasons, joint velocity and acceleration, if considered at all, would actually be computed numerically by first and second differences of the solution $w(t)$ obtained using Λ^{-1}. However, such numerical differentiation tends to amplify noise and introduce a delay. A different solution in which this transformation is not needed is presented next.

3.9.3 Resolved motion rate control (RMRC)

RMRC, or the Inverse Jacobian Cartesian control [13], is essentially a closed loop control in which the sensed joint position is immediately transformed by means of the kinematic equations Λ into the equivalent Cartesian position[13]. As shown in Figure 3.12 this Cartesian description (converted to its equivalent six-entry vector $w(t)$) is then compared to the desired Cartesian position $w_d(t)$ to form an error $\delta e(t)$ in Cartesian space.

This error, which may be presumed small under the action of the controller, may be mapped into a small displacement in joint space by means of the inverse Jacobian. These differential errors in joint space, essentially the derivative of the joint vector, are then multiplied by a diagonal gain matrix and fed into the velocity servos that control each joint [32]. Alternatively, a straightforward velocity loop topology might be used where the error is directly the difference from a Cartesian velocity setpoint and the measured Cartesian velocity [7]. The latter would be determined from a joint velocity sensing device (e.g., a tachometer), which is then converted to the hand velocity by means of the Jacobian. The hardware available will dictate the approach implemented. While the one-to-many Cartesian-to-joint-space trajectory conversion process implemented in Section 3.9.2 is replaced by some kind of one-to-one

[13] Recall that, as opposed to Λ^{-1}, this is not an ill-defined mapping but a one-to-one transformation.

coordinate conversion (Λ and J) inside the servo loop, the RMRC carries a heavy computational burden due to these kinematic transformations which need to be executed at servo rate. This drawback, common to other Cartesian-based control methods, like the Resolved Motion Acceleration Control (RMAC) [33], would, in general, degrade the stability and disturbance rejection capabilities of the system compared to joint-based systems.

3.10 Summary and Discussion

This Chapter has reviewed the most common forms of robot control in use today with the aim to set forth the fundamental control framework on which the approach examined in this dissertation builds upon. The straightforward linear independent-joint position servos implemented by most current robot manipulators work relatively well in many elementary industrial tasks and are, at present, more reliable and maintainable than multivariable methods. However, the performance of such controllers decreases rapidly when dynamic effects become significant (e.g., at high speeds or with varying loads). Hence, if the assumption of linearity in the manipulator is removed, this technique behaves satisfactorily only over a limited range of operation.

Dynamic control has produced model-based non-linear dynamic decoupling techniques and adaptive control methodologies that can, ideally, overcome such drawbacks. The potential advantages of VSC as a robust method for manipulator control has also been stated. However, the more and more stringent demands of complex MIMO systems, such as high-performance aerospace vehicles or advanced robotic systems, have prompted the seeking of alternative methodologies for their associated control systems. That is, control strategies which can naturally lead towards a system which can be regarded as the best possible system to accomplish a desired task, with respect to a standard performance index (e.g., time). In other words, optimal control. A survey of the two classical optimal control approaches to the non-linear MIMO time-varying TPBV problem being investigated in this work, the MP and DP, has been presented, along with an optimal technique specific for linear plants, the LQR controller. It has been made clear from the expositions on these Sections that most of the published work on this subject is at high mathematical levels, beyond the reach of applicability to real control problems. In fact, numerical solutions are normally the only option even for simple control problems. However, the "natural" aim of obtaining a "better" system will lead in the following Chapters to review this theory and propose a compromise between optimality and applicability as an approach to develop feasible optimal control strategies.

References

[1] Ogata K. *Modern Control Engineering.* Prentice-Hall Inc., second edition, (1990).

[2] Dorf R.C. *Modern Control Systems.* Addison-Wesley Publishing Company Inc., fourth edition, (1986).

[3] Brady M., Hollerbach J.M., Johnson T.L., Lozano-Perez T., and Mason M.T. *Robot Motion: planning and control.* MIT Press, (1982).

[4] Schilling R.J. *Fundamentals of Robotics.* Prentice-Hall Inc., (1990).

[5] Unimation (Europe) Ltd. *PUMA Robot Technical Manual,* (1982).

[6] CRS Plus Inc. *CRS A100/A200 Series Small Industriol Robot Systems Technical Manual,* (1990).

[7] Fu K.S., Gonzalez R.C., and Lee C.S.G. *Robotics: control, sensing, vision, and intelligence.* McGraw-Hill Book Co., (1987).

[8] Chestnut H. *Systems Engineering Tools.* John Wiley & Sons Inc., (1965).

[9] Franklin G.F., Powell J.D., and Emami-Naeini A. *Feedback Control of Dynamic Systems.* Addison-Wesley Publishing Company Inc., (1986).

[10] Phillips C.L. and Harbor R.D. *Feedback Control Systems.* Prentice-Hall Inc., third edition, (1996).

[11] Arimoto S. and Miyazaki F. Stability and robustness of pid feedback control for robot manipulators of sensory capability. In *Proceedings of the First International Symposium on Robotics Research*, pages 783–801. MIT Press, August (1984).

[12] Aström K.J. and Wittenmark B. *Computer Controlled Systems: theory and design.* Prentice-Hall Inc., (1984).

[13] Craig J.J. *Introduction to Robotics: mechanics and control.* Addison-Wesley Publishing Company Inc., second edition, (1989).

[14] Freund E. Fast nonlinear control with arbitrary pole-placement for industrial robots and manipulators. *International Journal of Robotics Research*, **1**(1):65–78, (1982).

[15] McKerrow P.J. *Introduction to Robotics.* Addison-Wesley Publishing Company Inc., (1991).

[16] Slotine J.J. and Li W. On the adaptive control of robot manipulators. *International Journal of Robotics Research*, **6**(3):49–59, (1987).

[17] Dubowsky S. and DesForges D.T. The application of model-referenced adaptive control to robotic manipulators. *Journal of Dynamic Systems, Measurement, and Control*, **101**:193–200, (1979).

[18] Koivo A.J. and Guo T.H. Adaptive linear controller for robotic manipulators. *IEEE Transactions on Automatic Control*, **AC-28**(1):162–171, (1983).

[19] Lee C.S.G and Chung M.J. An adaptive control strategy for mechanical manipulators. *IEEE Transactions on Automatic Control*, **AC-29**(9):837–840, (1984).

[20] An C.H., Atkeson C.G., and Hollerbach J.M. *Model-Based Control of a Robot Manipulator.* MIT Press, (1988).

[21] Young K.K.D. Controller design for a manipulator using theory of variable structure systems. *IEEE Transactions on Systems, Mon, and Cybernetics*, **SMC-8**(2):101–109, February (1978).

[22] Asada H. and Slotine J.E. *Robot Analysis and Control.* John Wiley & Sons, Inc., (1986).

[23] Pontryagin L.S., Boltyanskii V.G., Gamkrelidze R.V., and Mischenko E.F. *The Mathematical Theory of Optimal Processes.* Interscience Publishers Inc., (1962).

[24] Rozonoer L.I. L.s. pontryagin's maximum principle in the theory of optimum systems i, ii, iii. *Automation and Remote Control*, **20**(10-12):1288–1302,1405–1421,1517–1532, October/November/December (1959).

[25] Bellman R.E. *Dynamic Programming.* Princeton University Press, (1957).

[26] Kirk D.E. *Optimal Control Theory: on introduction.* Prentice-Hall Inc., (1970).

[27] Bryson Jr.A.E.. and Ho Y.C. *Applied Optimal Control: optimization, estimation, and control.* Blaisdell Publishing Company, (1969).

[28] McCausland I. *Introduction to Optimal Control.* John Wiley & Sons Inc., (1969).

[29] Eveleigh V.W. *Adaptive Control and Optimisation Techniques.* McGraw-Hill Inc., (1967).

[30] Shinners S.M. *Modern Control System Theory and Design.* John Wiley & Sons, Inc., (1992).

[31] Kalman R.E. Mathematical description of linear dynamical systems. *Journal SIAM Control*, series **A**:152–192, (1963).

[32] Klafter R.D., Chmielewski T.A., and Negin M. *Robotic Engineering. An Integrated Approach.* Prentice-Hall Inc., (1989).

[33] Luh J.Y.S., Walker M.W., and Paul R.P.C. Resolved-acceleration control of mechanical manipulators. *IEEE Transactions on Automatic Control*, **AC-25**(3):468–474, (1980).

Chapter 4

Survey of Alternative Robot Motion Schemes

4.1 Introduction

The concluding remarks drawn in the previous three Chapters have suggested that the manipulator trajectory planning and tracking problem could be conveniently regarded as tightly coupled if the rigid body dynamics of the robot are taken into consideration. This feature, along with advances in current processing technology, has naturally led to the consideration of optimal control methods in search of a solution to improve manipulator performance and/or meet more rigorous constraints.

As previously pointed out, this idea is not new and a number of researchers have attempted different methods to integrate dynamics with an IP (and maybe actuator limits) to obtain better trajectory/control solutions that can utilise the capabilities of the manipulator in full or nearly in full. This Chapter is a survey of results taken from the literature on optimal trajectory planning and control of robot manipulators. Some of the strategies reviewed will be for generic optimal control problems, but most concentrate on what is widely acknowledged as the primary measure of optimality, i.e., the time required by the manipulator to reach the desired location – also the focus of concern in this dissertation.

It should be noted that since the trajectory planning problem is reformulated as an optimal control problem with control and possibly state constraints, the resulting algorithms yield optimal/suboptimal trajectories along with an approximation [1] to the (generally) open-loop control torques that generate such trajectories. In view of this fact, the topic is referred to in the literature as either the optimal trajectory planning problem or the optimal control problem [2].

The fundamental nature of the different alternatives to the time-optimal robot motion problem has already suggested an initial division based on whether the dynamics are contemplated in the optimisation or not. A brief summary of several trajectory optimisation approaches which employ constant global estimates of the joint velocities, accelerations and maybe jerk limits is first described in Section 4.2. Given their geometric nature, the solutions proposed by these methods underlie, to a large extent, much of the trajectory planning theory described in Chapter 2. However, some kind of time-optimisation procedure - usually iterative in nature - is also added. When the minimum-time trajectory planning problem is further constrained by the details of the robot dynamics, a long-standing problem with vast appeal to finding practical schemes to improve robot motion performance emerges. Because of the importance of the problem, several approaches have been reported in the literature which have their roots in a paper published in 1971 by Kahn and Roth [1]. However, a rather simplistic classification imposed by the various levels of complexity of the given starting path is presented in Section 4.3. Finally, in Section 4.4, some comments are made regarding the general advantages and drawbacks of the different strategies presented and the reasons for opting for the approach undertaken in this dissertation.

[1] Exact manipulator dynamics are never fully known.

[2] Sometimes also as the optimal path planning problem, but the author believes this is misleading given the general remarks in Section 1.2.

4.2 Geometric Optimal Approach

This class of problems invariably assume an unconstrained desired motion given by the path end-points, and a set of intermediate knot points between path end-points. The trajectory through the knot points is therefore generated to ideally satisfy continuity and physical bounds. In addition, some sort of performance index is also (explicitly or implicitly) considered. Since dynamics are not examined here, the resulting optimal trajectory is then assumed to be followed by one of many well-known on-line tracking algorithms, like those described in Chapter 3, to drive the manipulator along the prescribed trajectory.

The trajectory suggested by Paul [2] can be essentially regarded as an extension of that described in Section 2.7.4, mostly focused on multi-point-to-point paths but applicable also to simple point-to-point trajectories with strategic mid knot points. It was based on a simple approach that eliminates stopping at each transition from one segment to another by looking one knot point ahead. The acceleration time to allow the manipulator velocity to change from maximum to minimum and vice versa was fixed, although smooth acceleration transitions were granted. A quartic polynomial was defined over the transitions between trajectory segments, whereas a linear polynomial, as the one used in the mid segment of Equation (2.13), sufficed after the transition. Further details of this approach can also be found in Paul's book [3]. An adaptation to Cartesian trajectory planning is also described there, which is shown to be conceptually simpler than the joint scheme but, as expected, computationally more expensive.

Luh and Lin [4] investigated the minimum-time path planning problem, where they derived a method for obtaining a time history of positions and velocities along a pre-specified path with a minimum travelling time under the constraints of Cartesian limits on linear and angular velocities and accelerations. A large-step gradient technique was attempted to shorten computing time, but its iterative nature still renders an off-line solution.

As described in Section 2.7.3, the addition of two extra knot points with unspecified joint displacements provided enough freedom for solving the trajectory planning problem under continuity conditions. This solution was adopted by Lin *et al* [5] where the transformed joint displacements for a number of pre-specified Cartesian points minus two were interpolated by piecewise cubic polynomials after adding the unknown two extra knots. The resulting spline functions were expressed in terms of time intervals between adjacent knots so that minimizing the total travelling time reduced to adjusting the time intervals between each pair of adjacent knots. In the paper, a non-linear iterative search algorithm was adopted which minimised the total travelling time by assuming the knot points to be the vertices of a flexible polyhedron. Basically, a new and better knot obtained in the search would replace the current worst knot to construct a better polyhedron for the next search. As a result, the flexible polyhedron will be moved closer to the optimal solution step by step. It is evident, however, that the calculations required to compute the optimum time-intervals and then spline the functions together need to be performed off-line.

Cao and Dodds [6] suggested, in a recent paper, a new approach for smooth and time-optimal joint trajectory planning by constraining the path defined by the knot points to a trajectory template based on Lin *et al*'s [5] piecewise cubic spline trajectory, along with a suitable objective function. The optimization process presented was divided into two phases. The first was to minimize the objective function by changing the positions of the joint knots in the piecewise cubic polynomial with initial time intervals suitably chosen so that a smooth path was obtained. A vector of Lagrangian multipliers (see Section 3.8.1) was introduced here to convert the constrained optimisation problem back into a linear unconstrained one which was then solved. The authors noted that the resulting linear systems could yield no solution or infinite solutions, yet a unique solution was assumed without proof. The second was to scale the time intervals for the time-optimal paths by contracting the travelling times so that the resulting joint velocities and accelerations or jerks at some knots of the paths were maximal with their limit constraints. This was again carried out iteratively by the One Dimensional search method until some minimum error boundary was achieved.

4.3 Dynamic Optimal Approach

The research reported in this group can be conceptually separated into three categories:

4.3.1 Motion constrained to specified geometric path

In the first method, the geometric path that the manipulator must follow is first assumed to be given in the form of a parameterised curve, $q_i = f(s)$, where q_i is the position of the i^{th} joint and S is the path parameter (see Section 1.2). This transformation reduces the $2n$ dimensional state space (position and velocity of each joint of an n DoF manipulator) to a two dimensional state (phase) space regardless of the number of joints in the robot. The time derivative of the parameter and the parameter itself completely describe the current state of the robot, and the constraint on input forces/torques is also converted to that of the path parameter by reducing it to bounds on the *pseudo-acceleration*, that is, the acceleration along the path. Hence, position, velocity and acceleration (thus torques) of the various joints are then related to one another through the parameterization of the path. Furthermore, given the value of the parameter along the path, its velocity and acceleration - if known, fully determines the input torques for all the joints. With this interpretation of the problem, the determination of the time-optimal solution becomes the selection of the pseudo-acceleration that produces the largest *pseudo-velocity* along the path. This is achieved by studying the effects that the constraints on the pseudo-acceleration (derived from the torques) impose on the values of the pseudo-velocity. The intersection of the regions determined by the pseudo-acceleration inequalities naturally lead to a region in the spatial pseudo-velocity *versus* spatial pseudo-acceleration phase plane which is referred to as the *admissible region* outside of which the phase trajectory must not stray.

Since it is very often necessary to specify the path the manipulator must follow (for example to avoid obstacles, or in welding, painting and like applications), several researchers have adopted this approach in the past. It is interesting to note that Shin and McKay [7], Bobrow *et al* [8] and Pfeiffer and Johanni [9] independently came to similar conclusions by following the same formulation, although their numerical algorithms and motivations were slightly different. Analogous non-standard numerical search procedures in the parameter phase plane have been employed by Shin and McKay [7] and Bobrow *et al* [8] to compute the maximum/minimum pseudo-acceleration along the path that produced the largest pseudo-velocity without violating the dynamic constraints. The authors note in their work that the new control variable - the pseudo-acceleration of the end-effector along the path - always takes the value on the bounds. It is interesting to note that in both papers the authors regard the problem as naturally suited for a solution in the language of optimal control with the employment of Pontryagin's MP (see Section 3.8.1). However, both discard it for the difficult closed-form or numerical solution on behalf of a simpler reasoning numerical algorithm.

An interesting addition to these methods with an experimental essence has recently been provided by Dahl [10] by proposing the idea of a path velocity controller (PVC) for modification of the velocity along a pre-specified path when the torques saturate. In his work, the path velocity controller acts as an outer feedback loop outside the ordinary robot controller, and modifies a nominal optimal velocity profile obtained according to [7] to achieve a reference trajectory which does not require more torque than is available (due, for instance, to disturbances or modelling errors). The computational overhead caused by the PVC is shown to be conceptually small and the necessary parameterisation of the main controller is also shown to be of the same complexity as the un-parameterised controller. However, experimental results are restricted to an oversimplified linear decoupled model of each joint of the manipulator, possibly for real-time computational reasons.

The same geometrical interpretation of the torque constraints leads to a similar search method confined by the field of acceleration/deceleration extremals in Pfeiffer and Johanni [9]. The approach, however, proved more illustrative and additionally, the technique of dynamic programming was also applied to optimize trajectories according to criteria other than minimum time. Although, as pointed out in Section 3.8.2, this technique is impractical for solving the path unconstrained TPBV problem as it would imply a search over $2n$ variables for an n DoF manipulator, if the path is specified the problem reduces to a search over a scalar parameter and its time derivative, thus minimising the "curse of dimensionality" referred to in Section 3.8.2. However, the numerical complexity still increases rapidly with respect to the (discrete) number of states. Another drawback of the application of DP to this problem is the non-smoothness of the trajectory due to the discrete grid representation. In addition, the DP approach does not offer any *a priori* insight into the structure of the optimal controller. Shin and Mckay also applied a dynamic programming search technique in another paper [11] by first discretising the phase-plane into a rectangular grid in which the cost of going from one point on the grid to the next spanned from the dynamic constraints of the manipulator and the performance index. This

algorithm was extended by Singh and Leu [12] to paths that did not need to be parameterised by a scalar parameter, but could simply be a sequence of points. The solution in this case was obtained in joint space by first solving for the inverse kinematics if the path points were initially specified in the hand coordinates. An additional constant constraint on joint velocities, independent of the available forces/torques, was imposed also to avoid instability at high speeds. The fact that the path is given enables the determination of the positions of all the other links if the position of one is known. Hence, the proposed algorithm reduced the problem to a search over the velocity of any one moving manipulator link. Although this ensured synchronisation of the links along the path, the optimality of the chosen "controlling" link was not proved. A recursive refinement scheme was employed to assure a faster convergence of the solution.

It was already noted when reviewing [7, 8] above, that the computational complexity of manipulator dynamics and the existence of state dependent constraints have always posed a major drawback in deriving a direct optimal solution to the (path constrained or not) TPBV problem by using Pontryagin's MP. However, the consideration of path constraints has only recently produced a solution to the problem as described in Shiller [13]. Rather than obtaining analytical closed-form solutions, a numerical solution is proposed with a gradient search that iterates over the initial value of one costate. The selection of this unknown is, essentially, a line optimization problem that is quite computationally inexpensive compared to the optimal DP search algorithms proposed previously. The solution presented, however, is still limited to low (2) DoF manipulators given the complexity of the solution. Chen and Chien [14] had previously employed a variation of Pontryagin's MP, the Extended Pontryagin's MP, which differs from the original only in the costate equations (see Equation (3.14)), to make some general remarks about the properties of the solution to the time-optimal control problem. They came to the conclusion that the existence of a time-optimal control requires either:

- One, and only one actuator be always in saturation on every finite time interval along the optimal trajectory.
- At least one of the actuators takes on values at limits if there exist singular critical points along the maximum velocity curve (points which represent a discontinuity in the pseudo-acceleration).

Altbough no results are given, a few existing numerical results from the literature reviewed here are cited to verify the theoretical results. These solutions just examined embrace a common factor with the majority of those described in Section 4.3.3 by proposing the direct use of Pontryagin's MP, although, as will be seen, no path is initially assumed given.

All these algorithms, though, rely heavily on an initial near-optimum path between end-points. Althougb some guidelines indicating how to generate these paths have been provided in Shin and McKay [15], the problem of specifying this path is very much an open problem for research.

4.3.2 Point-to-point path unconstrained motion with initial feasible trajectory

A more general approach to the truly minimum-time problem allows also for the shape of the path to be optimised in the process. This scheme combines the use of an initial feasible trajectory with an iterative numerical algorithm to determine the optimum parameters that minimise the path traversal time. Thus, the resulting optimal control problem, constrained by initial conditions, terminal conditions, control, state and the performance index is once more converted to a one-dimensional problem by parameterization with the proposed initial trajectory. Now though, a parameter optimisation technique is employed to search for the minimum-time path and trajectory. Uniform cubic B-spline polynomials have efficient computational properties and are 2-time differentiable functions, which make them an ideal candidate for smooth manipulator motions. These are splines (see definition 2.1) with the added properties of having the knots uniformly spaced in time and being equal on each time interval to a cubic polynomial composed in turn by four cubic basic functions. Hence, in approximating the angular displacement $q(t)$ with cubic B-splines, all the parameters involved in the problem become a function of the unknown coefficients of the cubic B-spline in S.

Polynomials of this form were employed by Bobrow [16] to represent the geometric path in the form of a set of spline vertices in a workspace containing obstacles. A parameter optimization technique was then used to search for a minimum-time path. For each iteration of path parameters, the time-optimal velocity profile along the path was obtained using the general-purpose non-linear constrained optimisation algorithm presented in [8]. A similar method was developed by Shiller and Dubowsky [17], but

additional manipulator end-effector constraints were also considered. As an alternative to parameterising the motion of each joint first in space (the spline parameter S) and then in time, Gilbert and Johnson [18] used B-splines to parameterise the motion of each joint directly as a function of time. Although similar in nature to the other methods described, the solution resulted in trajectories that were computationally less efficient. The solution proposed by Chen [19] was along the same lines. In this work, the numerical algorithm proposed to solve the resulting non-linear programming problem was an implementation of the sequential quadratic programming (SQP).

A parametric model in S based on a concatenation of cubic splines defined piecewise between given reference points was also employed by Wapenhans *et al* [20] for non-contact robot movements. The problems of minimum-time and a mixed cost function of minimum-time-energy were evaluated to determine the optimal path velocity along the path. The former was solved with an optimal algorithm that utilises the property described in [7] that an optimal solution must lie on the bounding surface of constraints in the phase plane. For the evaluation of the latter, the method of DP is used. The main novelty of the paper, though, is that it presents a complete procedure that encompasses both simulation and the implementation of a custom controller on an industrial manipulator. It is shown that for reasons of computational efficiency, dynamics equations are linearized around pre-determined points of the nominal trajectory. Furthermore, the output of the trajectory optimisation, i.e., the optimised trajectory reference values and the motor torques at each time interval, are calculated off-line before trajectory execution and employed in a feedforward controller (see Section 3.5.1) during execution.

Another proposition was presented by Kim and Shin [21] where the path was assumed to be given as a set of intermediate knot points. The algorithm was developed around the specific pre-requisite that an absolute tolerance in the path deviation at each corner point can be specified, allowing for the path to be optimized around the knot points. A set of local optimization problems - one at each corner point - was employed to optimize the problem, based on local upper bounds on joint acceleration derived from the manipulator dynamics.

The optimality of the majority of the solutions proposed under this scheme depends, however, on several factors, such as the initial guess for the unknown spline coefficients and, in some cases, the traveling time as well. The initial parameterisation of the problem with some piecewise polynomial spline allows for the infinite-dimensional optimal control problem to be approximated by a finite-dimensional one. Although this provides a way of finding an exact numerical solution, thus avoiding the complicated (and very often impossible) numerical integration of the manipulator dynamic equations of motion, the success of the approach still depends on how well the finite-dimensional problem can be solved. There is always a risk that the solution obtained may be a local minimum. However, above all, it depends on how well the exact solution can be approximated by uniform cubic B-splines or any other form of a polynomial spline trajectory. Obviously, this is not the case for all control laws, e.g. the class of bang-bang control laws.

In recent years several neural network (NN) architectures have been proposed to deal with the constrained optimisation problem presented here. This approach, presented for instance by Simon [22], overcomes, to some extent, the drawback that the path must fit some pre-specified form. However, in view of the many different network architectures which can be used for optimisation, the author stresses that the purpose of the approach is to demonstrate the applicability of NN's to the problem, rather than finding the best path planning network. While the paper presents a method to find the best interpolating curve with minimum joint jerk through an arbitrary set of knot and end-point constraints, there are no theoretical limitations to applying the method to other IP functions. The author also emphasises that an annealing-type technique, used in conjunction with the network to climb out of local minima, prevents the algorithm from being appropriate for real-time use, although it significantly improves the quality of the final solution by finding the best among many local minima solutions.

4.3.3 Point-to-point path unconstrained motion

The most general and truly optimal approach to trajectory planning addresses the category of robot manoeuvres in which the path of the end-effector is free, i.e., point-to-point unconstrained motions. This is particularly useful for specifying *gross motion* of the robot arm when it operates in a collision-free space. Otherwise, some sort of collision avoidance can be assumed at task level to specify appropriate collision-free control points. The manipulator control problem can then be addressed in a more general form in which the robot is given relative freedom to move along any trajectory between any two given

intermediate or end path points. Removing the set of path constraiots from the problem specification does not simplify the minimum-time problem as it might at first appear. While the optimal control problem for specified paths is a one-dimensional optimisation, optimising motions between given end-points has proved more difficult and computationally expensive because of the increased dimension of the problem. As indicated in the early solutions to the minimum-time problem in general, Pontryagin's MP is the natural choice for studying the optimal control of systems in which there is a constraint or limitation of some kind on the instantaneous value of the input signals. The minimum-time problem, subject to control bounds with no constraints on the path, can be considered as a special case of the more general optimal control problem. Moreover, if the two end-points are prescribed, the standard TPBV problem is obtained as outlined in Section 3.8.1. It was stated there that unless the system is of low order (first or second), time-invariant and linear, there is little hope of finding the solution analytically. Therefore, the highly non-linear manipulator equatioos of motion very often require the use of numerical methods to solve the problem. Along these lines, a number of algorithms have been proposed in the literature.

An extension of the gradient method based on an adjustable control-variation weight (ACW) matrix algorithm was used by Weinreb and Bryson [23] to indicate that most solutions tend towards bang-bang control. Because the ACW program, based on a steepest descent algorithm, was a continuous function optimization code, it was unable to achieve sharp discontinuities in the controls, and it was computationally intensive. A more efficient numerical optimization algorithm for small-size problems, the sequential gradient and restoration algorithm (SGRA), was employed by Lee [24] to obtain similar curves, also involving smooth transitions between control bounds. An interesting result with regards to the path unconstrained problem was obtained in this work which is in sharp contrast to the general remarks about the optimal solution to the path constrained problem made at around the same time by Chen and Chien [14]. Lee indicated that at least one of the two controls (since results were for a 2 DoF manipulator) was at its limits at any time instant, a result that seemed to be confirmed by other researchers. The switch time optimisation (STO) algorithm, proposed by Meier and Bryson [25], was also based on the steepest descent method for parameter optimization but, because it assumed that the trajectories are exact bang-bang solutions, it not only achieved sharp discontinuities in the controls but also large computational savings. The output of the program indicated that the bang-bang assumption was justified. Although very similar in nature to the point-to-point problem, Meier and Bryson concentrated efforts on minimising the time required to force a manipulator to travel a specified distance, i.e., open initial and terminal states. The results, however, are also applicable to the problem where initial and final states are specified. In Geering *et al* [26], a parameter optimisation algorithm was employed first to obtain the switching times for the bang-bang controls and an initial guess for the Lagrange multipliers. This was followed by a shooting method (originally proposed in [27]) for solving the TPBV problem for different initial conditions. It was found that in a singular arc, the corresponding control is not necessarily bang-bang. Fundamentally the same approach was described by Fotouhi-C and Szyszkowski [28], although the Forward-Backward Method (FBM) of integration was proposed to generate a sufficiently good guess for the initial costate vector and final time. A multiple-point shooting method was employed by Oberle [29] to obtain bang-bang and singular controls without using parameter optimization.

In all the aforementioned exact numerical approaches, the resulting time-optimal solutions are in open-loop form; i.e., the optimal controls are known as functions of time only, rather than as functions of the instantaneous manipulator state vector. Hence, they do not account for any unexpected disturbances which may act on the system. An alternative procedure was suggested in the key paper of Kahn and Roth [1], that of approximating the coupled non-linear equations of motion by a linear system around the final target point. The solution to the resulting linear system could therefore be obtained analytically, resulting in a suboptimal feedback control of the manipulator torques. However, this method is only valid when manipulator motion is restricted to a small region, since the linearized equations of motion would not be valid if the manipulator was located away from the final target point. The paper by Wiens and Berggren [30] was also based upon this idea. They applied optimal control theory to solve the unconstrained point-to-point motion problem, although their main concern was to minimise the energy and non-linear dynamic effects (inertia, centripetal and Coriolis forces) during the trajectory planning. A linearised inertia sensitivity index was used in the cost function to quantify the robot's non-linear behavior over the whole workspace. Hence, an algorithm formulated in terms of the TPBV problem was presented that kept the energy consumption under control while minimizing

the non-linearities. In order to overcome the difficulty associated with solving the non-linear TPBV problem, the system equations were linearised and decoupled in feedback form at the beginning of every controller sampling interval, following the idea of *averaged dynamics* first proposed by Kim and Shin [31] for the general time-fuel suboptimal control problem. This is a simple approach to linearise the manipulator dynamics around the current and end-state, hence allowing for a feedback controller to be implemented. The authors point out that although the calculation of the complex inertia sensitivity index had to be performed at each sample interval, thus relegating the algorithm to off-line use only, with the use of *a priori* knowledge of the robot's kinematic and dynamic models a map of the inertia sensitivity index over the whole workspace can be generated and used *a posteriori* for on-line use. Simplified linear and decoupled robot dynamics have been assumed by Krejnin *et al* [32] in order to avoid the complex solution of the TPBV problem. Simulation and implementation results on an industrial robot seem to confirm the assumption that large gear reduction ratios driving the manipulator effectively decouple the individual joints from one another. The solution, however, can not be made extensive to other manipulators, e.g., high inertial robots such as direct or semi-direct actuated robots.

Some alternative techniques to the formulation of the minimum-time problem as a TPBV problem according to Pontryagin's MP have also been presented in the literature. For example, a search technique, in many ways similar to DP, was attempted by Sahar and Hollerbach [33] which involved joint tesellation, a dynamic time-scaling algorithm and a graph search. A very simple example with large computational requirements was presented, and the authors recognised that the primary benefit of the solution would be as a means to provide a first idea of what the minimal-time path looks like. Another alternative technique, digital in nature, to the task of computing the state switching curves was presented in [34], motivated by deadbeat control. The method proposed a digital state feedback algorithm to compute the near-minimum trajectory by placing the poles of the closed-loop system, linearised around the final state, in the Z-plane, without violating the constraints on the actuator torques. A time-variant state-feedback gain matrix was employed to force the location of all the poles of the system towards the origin if the torque bounds are not violated, or inside the unit circle of the Z-plane such that all constraints are satisfied and at least one of the joint torques is equal to either its maximum or minimum value. The authors state that the algorithm yields the required feedback matrix off-line and that, despite being time-variant, it is assumed constant during each sampling period, due to its digital character.

4.4 Summary and Discussion

While the methods described above are attractive in that they yield optimal/suboptimal solutions to the robot motion problem, the great majority suffer from the same shortcomings: they result in impractically complicated schemes, most of them solved via iterative numerical algorithms, hence at a large computational expense. In fact, research in time-optimal trajectory planning and control to date has been demonstrated almost exclusively by simulation and very few authors have discussed implementation issues and presented experimentals results. Undoubtedly, simulation has an important place in the development of control algorithms, for example in the analysis of stability or in checking the proper use of numerical methods. However, the real world is hard to simulate and it is crucial that extensive experimental evaluation be made too. This is of particular importance if the claims of increasingly complex model-based modern control techniques for MIMO systems, such as optimal or adaptive control, are to be taken seriously and become routine in robotics practice. From the papers reviewed above, the few exceptions which include practical results correspond, understandably, to some of the most recent papers, i.e. [10, 13, 20, 32, 35] which demonstrated the merits of time optimal control, and showed the significant contribution that the often ignored motor dynamics have in the optimization process. However, practical limitations due to the dimensionality of the problem stated before, have restricted the work to motions along either explicitly specified paths or an initial feasible trajectory to obtain exact optimal solutions. Another practical issue, that a closer examination of the proposals show, is that the the vast majority of the trajectory planning schemes yield an optimal control strategy in an open-loop fashion. Hence, when it comes to actually tracking the desired optimal trajectory two apparent alternatives are at hand:

- The resulting trajectory is then followed using a feedback control tracking algorithm, which could or could not account for the effect of the dynamics. Since the trajectory has been designed to be

optimal and within physical and dynamic constraints, a proper controller design would guarantee the optimal motion.

- Employ the torques obtained from the dynamic equations of motion and the optimal trajectory as an approximation to the control action.

Due to modelling inaccuracies and unknown disturbances in the control of real manipulators, the actual optimal trajectory may deviate substantially from the planned trajectory if the control loop is not closed. A feedback controller is therefore needed to direct the manipulator back to the planned trajectory when deviation occurs. The advantage of planning a trajectory before actually controlling the plant, i.e. off-line, is that all non-linearities in the manipulator dynamics can be taken into account in the off-line computation of control forces/torques without overwhelming the control computer, as would be the case for on-line control with computation of manipulator dynamics such as using the computed torque control of Section 3.5.2. This solution has been invariably adopted in all the experimental results presented to date except [32]. Some implemented a simple PD controller [10, 13, 35], while others opted for a feedforward device to make use of the optimal forces/torques previously calculated off-line and saved in a look-up table [20].

The solution implemented in this dissertation studies the viability of the second alternative, in particular as applied to the point-to-point unconstrained problem. It is an attempt to perform optimal motions with respect to a timing performance index in an on-line fashion with regards to both trajectory and control. Furthermore, full rigid-body dynamics are taken into account, hence generalising the solution to a wide range of manipulators. The author, however, understands the serious hazards and limitations that a pure open-loop approach involves, and in view of this fact this work is loosely based on the optimal feedback controller idea initially described by Kim and Shin [31] and, specifically, the simple concept of averaged dynamics also employed by Wiens and Berggren [30]. The process utilises all available dynamic information of the current (sampled) and the final states to update the dynamics continually at each sample interval, implicitly compensating, to some extent, for dynamic approximation errors. The update of the manipulator dynamics at each sample interval and the averaged dynamics are both simple, and therefore considered suitable for real-time implementation. In addition to adopting this idea, the work described in this dissertation explores further alternatives towards an efficient on-line implementation of the algorithm. In particular, a close look at the manipulator electro-mechanical mechanisms and an enhanced general form of the dynamics linearisation approach provides a more realistic solution to the design of a truly on-line near-optimal unconstrained trajectory generator and controller. With these issues in mind, a practical algorithm for the general point-to-point unconstrained motion based on an elementary dynamics linearisation method is developed for the on-line near-time-optimal `feedback` control of robot manipulators. The resulting TPBV problem in joint space has been solved analytically in real-time using Pontryagin's MP formulation and assuming bang-bang control for each robot joint.

Sundar and Shiller [36] recently stated in their paper that:

> *"To date (1996), no practical method has been developed for the on-line time-optimal feedback control of manipulators with non-linear dynamics."*

The work undertaken in this thesis is an attempt to fulfill that goal within the limits imposed by current technology.

References

[1] Kahn M.E. and Roth B. The near-minimum-time control of open-loop articulated kinematic chains. *Journal of Dynamic Systems, Measurement, and Control. Transactions of the ASME*, **93**(3):164–172, September (1971).

[2] Paul R.P. The mathematics of computer control manipulator. In *Proceedings of the Joint Automatic Control Conference*, volume **1**, pages 124–131, (1977).

[3] Paul R.P. *Robot Manipulators. Mathematics, Programming, and Control.* MIT Press, (1981).

[4] Luh J.Y.S. and Lin C.S. Optimum path planning for mechanical manipulators. *Journal of Dynamic Systems, Measurement, and Control. Transactions of the ASME*, **103**:142–151, June (1981).

[5] Lin C.S., Chang P.R., and Luh J.Y.S. Formulation and optimization of cubic polynomial joint trajectories for industrial robots. *IEEE Transactions on Automatic Control*, **AC-28**(12):1066–1073, December (1983).

[6] Cao B. and Dodds G.I. Time-optimal and smooth joint path generation for robot manipulators. In *Proceedings of Control'94*, pages 1122–1127, March (1994).

[7] Shin K.G. and McKay N.D. Minimum-time control of robotic manipulators with geometric path constraints. *IEEE Transactions on Automatic Control*, **AC-30**(6):531–541, June (1985).

[8] Bobrow J.E., Dubowsky S., and Gibson J.S. Time-optimal control of robotic manipulators along specified paths. *International Journal of Robotics Research*, **4**(3):3–17, Fall (1985).

[9] Pfeiffer F. and Johanni R. A concept for manipulator trajectory planning. *IEEE Journal of Robotics and Automation*, **RA-3**(2):115–123, April (1987).

[10] Dahl O. Path-constrained robot control with limited torque - experimental evaluation. *IEEE Transactions on Robotics and Automation*, **10**(5):658–669, October (1994).

[11] Shin K.G. and McKay N.D. A dynamic programming approach to trajectory planning of robotic manipulators. *IEEE Transactions on Automatic Control*, **AC-31**(6):491–500, June (1986).

[12] Singh S. and Leu M.C. Optimal trajectory generation for robotic manipulators using dynamic programming. *Journal of Dynamic Systems, Measurement, and Control. Transactions of the ASME*, **109**:88–96, June (1987).

[13] Shiller Z. Time-energy optimal control of articulated systems with geometric path constraints. *Journal of Dynamic Systems, Measurement, and Control. Transactions of the ASME*, **118**:139–143, March (1996).

[14] Chen Y. and Chien S.Y.P. General structure of time-optimal control of robotics manipulators moving along prescribed paths. In *Proceedings of the American Control Conference*, volume **2**, pages 1510–1514, March/April (1992).

[15] Shin K.G. and McKay N.D. Selection of near-minimum time geometric paths for robotic manipulators. *IEEE Transactions on Automatic Control*, **AC-31**(6):501–511, June (1985).

[16] Bobrow J.E. Optimal robot path planning using the minimum-time criterion. *IEEE Journal of Robotics and Automation*, **4**(4):443–450, August (1988).

[17] Shiller Z. and Dubowsky S. Robot path planning with obstacles, actuator, gripper and payload constraints. *The International Journal of Robotics Research*, **8**(6):3–18, December (1989).

[18] Gilbert E.G. and Johnson D.W. Distance functions and their application to robot path planning in the presence of obstacles. *IEEE Journal of Robotics and Automation*, **RA-1**(1):21–30, March (1985).

[19] Chen Y.C. Solving robot trajectory planning problems with uniform cubic b-splines. *Optimal Control Applications & Methods*, **2**:247–262, (1991).

[20] Wapenhans H., Hölzl J., Steinle J, and Pfeiffer F. Optimal trajectory planning with application to industrial robots. *The International Journal of Advanced Manufacturing Technology*, **9**:49–55, (1994).

[21] Kim B.K. and Shin K.G. Minimum-time path planning for robot arms and their dynamics. *IEEE Transactions on Systems, Man and Cybernetics*, **SMC-15**(2):213–223, March/April (1985).

[22] Simon D. The application of neural networks to optimal robot trajectory planning. *Robotics and Autonomous Systems*, **11**:23–34, (1993).

[23] Weinreb A. and Bryson Jr.A.E. Optimal control of systems with hard control bounds. *IEEE Transactions on Automatic Control*, **AC-30**(11):1135–1138, November (1985).

[24] Lee A.Y. Solving constrained minimum-time robot problems using the sequential gradient restoration algorithm. *Optimal Control Applications & Methods*, **13**:145–154, (1992).

[25] Meier E. and Bryson Jr.A.E. Efficient algorithm for time-optimal control of a two-link manipulator. *Journal of Guidance Navigation and Control*, **13**(5):859–866, September/October (1990).

[26] Geering H.P., Guzella L., Hepner S.A.R., and Onder C.H. Time-optimal motions of robots in assembly tasks. *IEEE Transactions on Automatic Control*, **AC-31**(6):512–518, June (1986).

[27] Lastman G.J. A sbooting method for solving two-point boundary-value problems arising from non-singular bang-bang optimal control problems. *International Journal of Control*, **27**(4):513–524, (1978).

[28] Fotouhi-C R. and Szyszkowski W. A numerical approach for time-optimal control of double arms robot. In *Proceedings of the 4th IEEE Conference on Control Applications*, Albany, USA, September (1995).

[29] Oberle H.J. Numerical computation of singular control functions for a two-link robot arm. In *Proceedings of the Conference on Optimal Control and Variational Calculus*, pages 244–253, Oberwolfach, FRG, (1986).

[30] Wiens G.J. and Berggren M.J. Suboptimal path planning of robots: minimal nonlinear forces and energy. *Journal of Dynamic Systems, Measurement, and Control. Transactions of the ASME*, **113**:748–752, December (1991).

[31] Kim B.K. and Shin K.G. Suboptimal control of industrial manipulators with a weighted minimum time-fuel criterion. *IEEE Transactions on Automatic Control*, **AC-30**(1):1–10, January (1985).

[32] Krejnin G.V., Sattar T.P., and Smelov L.A. Towards increasing the speed of manipulation mechanisms. *Journal of Machinery Manufacture and Reliability*, **3**:62–68, (1994).

[33] Sahar G. and Hollerbach J.M. Planning of minimum-time trajectories for robot arms. *International Journal of Robotics Research*, **5**(3):90–100, Fall (1986).

[34] Kao C.K., Sinha A., and Mahalanabis A.K. A digital algorithm for near minimum-time control of robot manipulators. *Journal of Dynamic Systems, Measurement, and Control. Transactions of the ASME*, **109**:320–327, December (1987).

[35] Shiller Z., Chang H., and Woug V. The practical implementation of time-optimal control for robotic manipulators. *Journal of Robotics & Computer-Integrated Manufacturing*, **12**(1):29–39, (1996).

[36] Sundar S. and Shiller Z. A generalized sufficient condition for time-optimal control. *Journal of Dynamic Systems, Measurement, and Control. Transactions of the ASME*, **118**:393–396, June (1996).

Chapter 5

Dynamic Modelling and Simulation

5.1 Introduction

This Chapter examines what might be regarded as the first step in any controller design – the accurate modelling of the plant to be controlled. This is a preliminary stage to the design of the optimal control solution adopted, which is described in the following Chapter. The importance of manipulator dynamics stems not only from its use in model-based control strategies, such as those described in Chapter 3, but also in simulation and analysis. It has been stated that, in practice, this modelling step may occupy greater than 80-90% of the effort required in control systems analysis and design [1, 2]. Consequently, the significance of understanding the dynamic structure of the plant is found to be particularly important, and the framework is quite detailed.

Since the mathematical representation of the dynamic behaviour of a general robot manipulator can be complex, the modelling of the mechanical structure and that of the joint driving motors have been studied separately: in Section 5.3, the mechanical equations of motion of the robot device, under the assumption (taken from the physical robot available for testing) of n rigid links joined together serially by n revolute joints, are derived. The decision to opt for the Lagrangian formulation of mechanics is also discussed, whilst the actuator characteristics of the DC servomotor driving each robot joint are described in Section 5.4. Although the dynamics of the joint actuators are, very often, overlooked in the manipulator controller design, the inability to specify joint torques in the typical servomotors of commercial arms would deem most advanced control strategies unsuitable, since almost all of them are invariably based on the capability to control joint torques. Hence, modelling the actuators is not only aimed at accurately simulating the most significant motor and driver dynamics, but to allow driving the actuators in torque mode, as specified by the controller design. It is, however, not enough to come up with a nominal model structure for the robot arm, and the validation procedure followed to measure the reliability of both models is outlined in Sections 5.3.4 and 5.4.3 for the mechanical and electro-mechanical models respectively.

The rapid development of computer hardware and graphics software during the last decade has added a new dimension to systems study through modelling and simulation. Features such as colour graphics, solid 3D objects and animations have been incorporated to give way to graphical programming as the natural approach to plan complex robot motion safely, quickly and easily. The potential major role that advanced computer graphics are starting to play in the practice of (robot) modelling and simulation, and in particular its important contribution to the simulation of the novel control algorithms presented in this thesis, is outlined in Section 5.5. Finally, in Section 5.6, some concluding remarks are extracted from the results presented in this Chapter.

5.2 System Modelling

There are many conceptually different ways in which the components of the mechanical manipulator that need to be modelled, namely the motor characteristics, the kinematic parameters and the inertial (and maybe load) parameters, can be obtained.

Both kinematic and inertial characteristics can be determined from design blueprints. Because machining has limits of precision and assemhly may be imperfect, there will always be slight variations in

kinematic parameters – link lengths that are a little off nominal values or neighbouring joint axes that are not quite so parallel. Since striving for even greater precision can be costly, "kinematic calibration" has come to be recognised as a necessary process for any robot to avoid the resulting tip inaccuracies and miss-locations of the robot with respect to external reference frames [1].

The geometric information, in the form of Computer Aided Design (CAD) models of the parts, can be further combined with specific densities of materials to estimate the inertial parameters. This approach requires intensive human involvement and is, as before, subject to modelling errors. Link inertias can also be determined experimentally by disassembling the robot, and then weighting the pieces for mass, counterbalancing for center of mass, and swinging for moments of inertia [3]. In addition to requiring intensive human involvement, as in the previous method, this procedure introduces considerable measurement difficulties.

Another approach, applicable to the characterisation of all three types of unknown parameters, is *System Identification*, i.e., the determination of parameters values from the analysis of input/output sensory data. The particular class of system identification that estimates parameters of a known model structure is termed *Parameter Estimation*. In general, the system identification approach sacrifices, for convenience, the precision that can be obtained by some of the above techniques, since minimal human involvement is required. Various system identification approaches have been suggested and a good review is provided in [1, 4].

Ultimately, the easiest way to obtain the robot parameters is from the manufacturer. This is, however, an unusual situation because the information is often unknown even by the robot manufacturers themselves. Furthermore, provided that restricted information is known, they are normally unwilling to make it widely available, both for proprietary reasons and also because of safety issues. For the work presented here an agreement was signed with the commercial robot manufacturer, CRS Robotics Corporation [1]. As a result, a large number of the parameters necessary to model the robot arm were provided. This meant that the physical laws and relationships that described the system could be exploited to obtain the model structure. Kinematic and dynamic parameters, as well as most of the motor characteristics, were supplied, to which the structure of the dynamic equations of motions and motor model described in this Chapter were accommodated. Although some calibration would have been needed for precise positioning, because the control solution developed in this work is focused on *gross motion*, these unavoidable miss-alignments and imperfections were not of much concern here and the parameters were incorporated into the model structures for validation.

5.3 Mechanical Modelling

There are a number of procedures for generating the dynamic equations of motion for a robot arm, i.e., the equations which relate joint forces and torques [2] $\tau(t)$ to positions $\theta(t)$, velocities $\dot{\theta}(t)$ and accelerations $\ddot{\theta}(t)$, in terms of the specified kinematic and inertial parameters of the links. At present, a number of ways have been proposed for this purpose, such as:

- Lagrange-Euler(L-E) method [5]
- Newton-Euler(N-E) method [6]
- Recursive Lagrangian method [7]
- Kane's method [8]
- Appel's method [8]
- Generalised D'Alambert principle method [6]

In one form or another, the models are obtained from known physical laws such as the law of Newtonian mechanics and Lagrangian mechanics. These methods are "equivalent" to each other in the sense that they describe the dynamic behaviour of the same physical robot manipulator. However, the structure of these equations and, particularly, the computational efficiency of the equations may

[1] 5344 John Lucas Drive, Burlington, Ontario, Canada L7L 6A6.

[2] Since the class of robotic manipulators considered in this work is assumed to have only revolute joints, from now on only torques will be referred to without loss of generality.

differ, as they are obtained for various reasons and purposes, such as suitability for simulation, real-time control, parameter significance, controller design, etc.

Among these methods, the L-E and the N-E formulation have been generally used. These methods, based on the Lagrangian and Newtonian mechanics respectively, have their own advantages and disadvantages. The advantage of the N-E method, which might be said to be a "force balance" approach to dynamics, is that the amount of computation necessary to obtain the joint generalised torques is quite small. On the other hand, it is messy to derive and difficult to apply to the design of motion control strategies of the robot manipulator because it is a recursive algorithm and, consequently, no insight into the structure of the equations is provided. In order to design control strategies and to perform dynamic simulations, an explicit set of closed form differential equations in state-space form is often useful, so that the coupling reaction torques can be easily identified.

The L-E formulation of mechanism dynamics, which might be said to be an "energy approach" to dynamics, is a relatively simple, elegant approach which yields a set of differential equations in symbolic form where the physical meaning of each term in the equations is clear. The most significant drawback of the L-E formulation arises from the computational inefficiency of its general form, which has traditionally been a bottleneck for model-based control. However, as was already outlined in Section 3.8 – where optimal control was introduced, and will again be discussed in Chapter 6 – when the specific controller is designed, a single closed-form state-space expression of the plant dynamics is a requisite for the convenient analysis and design of the optimal controller, and in general any model-based controller. Hence, the L-E formulation will be the preferred dynamic model in the design of the motion controller.

5.3.0.1 A note about the N-E and L-E computational issue

The author would like to emphasise a couple of observations with regards to the form and computational efficiency of these two common formulations which further support the decision to employ the L-E dynamics approach:

1. The N-E formulation provides a computational scheme as a set of recursive equations in which a backward recursion propagates the kinematic information, including velocities and accelerations, and the forward recursion propagates the torques exerted on each link of the robot. Although, as previously stated, these equations result in an efficient numerical computational algorithm applicable to any robot, it is interesting to note that they can be further developed analytically to obtain the same set of closed form symbolic equation on $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$ which is directly obtained with the L-E formulation. An example can be found in Craig [9]. For the general derivation of this additional transformation, presented in Brady *et al* [10], the closed form N-E equation is shown to be of the same complexity as that of the L-E, $O(n^4)$, while the work proposed by Hollerbach [7], also reviewed in [10], shows how the L-E equations can be recast into an $O(n)$ recursive form identical with the N-E formulation. Therefore, it is concluded that the emphasis on computational complexity should rest on the structure of the computation rather than on the derivation from the Lagrange or the Newton-Euler formulation.

2. While it is true that the N-E iterative scheme is quite efficient as a *general* means of computing the dynamics of any manipulator, several authors have published articles showing that for any given manipulator, customised closed form dynamics are more efficient than even the best of the general schemes [9, 11]. The reasoning behind this is that a fair amount of computational load can be eluded by eliminating operations with zero-valued factors or reducing the number of operations, either by gathering common factors or by introducing simplifying trigonometric substitutions, or both. For a good review on current automatic symbolic manipulation tools, the reader may refer to [12].

The treatment of robot arm dynamics presented here is patterned primarily after discussions found in Fu *et al* [6], Brady *et al* [10], Craig [9], Sanders [13] and Rivin [5], to which the reader may refer for further details. It is out of the scope of this brief introduction about manipulator dynamics modelling and, in general, the work presented in this dissertation, to explore the details of how the different derivations of robot dynamics are carried out, but to find a suitable scheme appropriate for the control problem being studied. Thus, no further insight into the general problem of robot arm dynamics will be provided and the focus of attention herein will be on deriving the L-E formulation of the CRS

A251 general purpose industrial manipulator. The reader may refer to any of the above references or [8, I4, 15] for more details on other alternatives. It will become apparent in the following Chapter how, despite customising in this work the equations of motion of the manipulator available for testing for computational efficiency, the controller is designed for the general form of the closed-form dynamic equations of motion of a robotic arm, thus generalising the algorithm to practically any manipulator.

5.3.1 Lagrangian formulation of the equations of motion

The derivation of the dynamic equations of an n DoF manipulator (excluding the dynamics of the electronic actuating devices, described in Section 5.4) is based on the understanding of the Lagrange-Euler energy equation, described as

$$\tau_i = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} \qquad \text{for } i = 1 \ldots n \qquad (5.1)$$

where L, the Lagrangian function, is defined as the Kinetic energy minus the Potential energy of the manipulator, that is, $L = K - P$.

It is therefore necessary to properly choose a set of generalised coordinates q_i to describe the system. Generalised coordinates are used as a convenient set of coordinates which completely describe the location (position and orientation) of objects in the mechanical system with respect to the reference coordinate system (usually located at the fixed base of the robot). For a manipulator with rotary joints the angular positions of the link joints, θ, provide a natural choice for the generalised coordinates, q, because they are readily available from sensing devices like encoders and potentiometers. Hence, they will be used interchangeably hereafter.

As shown in Equation (5.1), the L-E formulation requires knowledge of the kinetic energy, K, of the physical system, which in turns requires knowledge of the velocity of each joint. Likewise, gravity loading effects are accounted for in the potential energy P. In both cases, manipulator's kinematic parameters are required in the derivation. Having obtained the Lagrangian function L of the arm, the L-E formulation can then he applied which yields the necessary torque to drive the manipulator in a matrix-vector equation form as:

$$\boldsymbol{D}(q)\ddot{q} + \boldsymbol{H}(q,\dot{q}) + \boldsymbol{G}(q) = \tau \qquad (5.2)$$

where $\boldsymbol{D}(q) \in \Re^n \times \Re^n$ denotes the inertia matrix associated with the distribution of mass, $\boldsymbol{H}(q,\dot{q}) \in \Re^n \times \Re^1$ is a vector containing all interaxial velocity-dependent coupling terms arising from centripetal and Coriolis forces and $\boldsymbol{G}(q) \in \Re^n \times \Re^1$ represents the gravity force terms. Joint torques are included in vector $\tau \in \Re^n \times \Re^1$.

Equation (5.2) is a highly coupled, non-linear, second order symbolic differential equation. The dynamic equation is a closed form expression, in the sense that the dependence of a joint torque on movements at all joints is made explicit. In the above general expression in closed form, most terms are re-evaluated many times, which is the main reason for the inefficiency of this formulation, and the reason why recursive forms of the dynamics have been proposed to avoid this duplication (see, for instance, Hollerbach [7]). Moreover, the common use of $\Re^4 \times \Re^4$ homogeneous coordinate transformation matrices [3] to describe the spatial relationship between neighbouring link coordinate frames in Equation (5.2), further limits the computational efficiency of the formulation, despite resulting in a convenient and compact algorithmic description of the manipulator equations of motion.

In what follows, an expression for the expanded terms that form the closed form Equation (5.2) for the CRS A251 robot manipulator is presented. A customised closed form of the dynamics has been developed, which is derived from the vector representation of the Cartesian coordinates of a point-mass inertial distribution for each link, as can be seen in the kinematic sketch of the robotic arm shown in Figure 5.1, thus avoiding the inefficient use of homogeneous matrices. This simplifying mass-point approximation of the link inertias in the design of the robot equations of motion was imposed by the structural and dynamic analysis information provided by the manufacturer, which assumed, for some links, a single point-mass at the inertial center of gravity, whereas others were described by two mass-point inertial locations. In any case, this dynamically simple approximation to the manipulator mass distribution implies that the inertia tensor at the center of mass for each link is the zero matrix [9].

[3] Normally based on the standard Denavit-Hartenberg link coordinate representation [6].

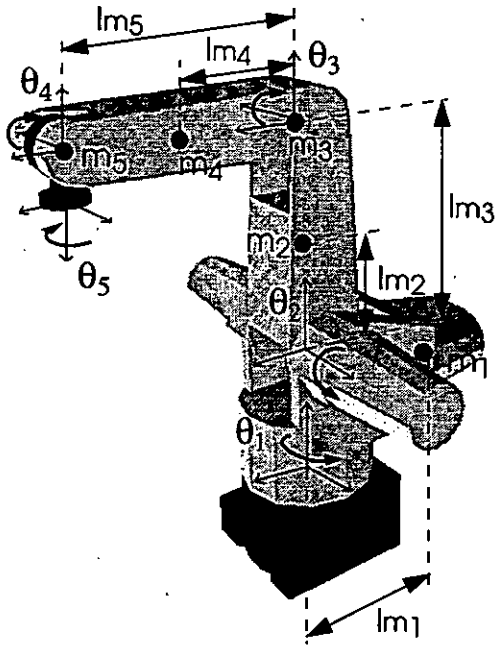

Figure 5.1: 3D model of the CRS A251 industrial manipulator with point mass approximations.

Joint	Range
θ_1	$\{\pm 175°\}$
θ_2	$\{-110°, 0°\}$
θ_3	$\{-40°, 90°\}$
θ_4	$\{\pm 110°\}$
θ_5	$\{\pm 180°\}$

Table 5.1: CRS A251 workspace.

Because the link products of inertia are difficult to measure and are frequently taken to be zero, based on assumed symmetry, this approximation is not so severe, and validation results shown in Section 5.3.4 have confirmed this assumption.

The arm is constructed of high tensile aluminium alloy components. It features side panels held rigid by crossmembers and linked by a stressed aluminium skin. This construction technique gives light weight while contributing to the rigidity of the arm, which in part allows for the high speed and accuracy of the system. Since the gravity forces are not counterbalanced, motors for vertical joints are equipped with automatic brakes to prevent the collapse of the manipulator configuration if the power supply to the joint motors is interrupted. As seen in Figure 5.1, the CRS A251 robot system is a 5 DoF open-chain articulated arm: the waist (θ_1), shoulder or upper-arm (θ_2), elbow or fore-arm (θ_3), wrist bend - or pitch (θ_4) and wrist roll (θ_5). The following analysis, however, relates only to the principal manipulator structures performing regional or *gross* motions (major linkage). The orientation links (wrist) and possible tooling devices (end-effectors) will be considered as single inertias. This is so because, in comparison with the other links, wrist joints are usually dominated by inertias, with gravity and inertial coupling effects in the range of one or two orders of magnitude down [12]. Therefore, the gross motion links considered for optimisation hereafter are the two main movable links in the gravitational field - upper-arm and fore-arm, and the robot waist.

The range of motion for each joint is presented in Table 5.1. This has been chosen according to the range of the encoder pulses for each motor joint instead of the range employed by the industrial controller [4], because the control will be later implemented at the pulse level for simplicity. Thus, the robot kinematic configuration shown in Figure 5.1 corresponds to $(0°, -90°, 0°, -90°, 0°)$, for joints $i = 1, ..., 5$.

Since the customised closed form equations of the dynamics of the CRS A251 arm are quite involved,

[4] Although, of course, the absolute rotational range is the same.

Parameter	Description	Value	Units (S.I.)
m_1	point mass	4.35	kg
m_2	point mass	0.84	kg
m_3	point mass	0.89	kg
m_4	point mass	0.62	kg
m_5	point mass [5]	0.59	kg
lm_1	m_1 radii	0.135	m
lm_2	m_2 radii	0.114	m
lm_3	m_3 radii	0.254	m
lm_4	m_4 radii	0.127	m
lm_5	m_5 radii	0.254	m

Table 5.2: CRS A251 kinematic and dynamic characteristics.

only the expressions for the torques are presented next. The derivation of these symbolic expressions can be found in Appendix A. To simplify the equations, the following standard notation is used:

$$s_i = sin(\theta_i) \quad c_i = cos(\theta_i) \quad s_{ij} = sin(\theta_i + \theta_j) \quad c_{ij} = cos(\theta_i + \theta_j)$$

The equations of motion are then found to be:

$$\begin{aligned}
\tau_1 &= [\{m_1 lm_1^2 + (m_2 lm_2^2 + (m_3 + m_4 + m_5) lm_3^2) c_2^2 + (m_4 lm_4^2 + m_5 lm_5^2) cos_3^2 \\
&\quad + 2lm_3 (m_4 lm_4 + m_5 lm_5) c_2 c_3\} \ddot{\theta}_1] \\
&\quad + [-2\{(m_2 lm_2^2 + (m_3 + m_4 + m_5) lm_3^2) s_2 c_2 + (m_4 lm_4 + m_5 lm_5) lm_3 s_2 c_3\} \dot{\theta}_1 \dot{\theta}_2 \\
&\quad - 2\{(m_4 lm_4^2 + m_5 lm_5^2) s_3 c_3 + (m_4 lm_4 + m_5 lm_5) lm_3 s_3 c_2\} \dot{\theta}_1 \dot{\theta}_3] \\
\tau_2 &= [\{m_2 lm_2^2 + (m_3 + m_4 + m_5) lm_3^2\} \ddot{\theta}_2 - \{lm_3 (m_4 lm_4 + m_5 lm_5) c_{23}\} \ddot{\theta}_3] \\
&\quad + [\{(m_2 lm_2^2 + (m_3 + m_4 + m_5) lm_3^2) s_2 c_2 + (m_4 lm_4 + m_5 lm_5) lm_3 s_2 c_3\} \dot{\theta}_1^2 \\
&\quad + \{(m_4 lm_4 + m_5 lm_5) lm_3 s_{23}\} \dot{\theta}_3^2] \\
&\quad - [\{(m_2 lm_2 + (m_3 + m_4 + m_5) lm_3) c_2\} g] \\
\tau_3 &= [-\{lm_3 (m_4 lm_4 + m_5 lm_5) c_{23}\} \ddot{\theta}_2 + \{m_4 lm_4^2 + m_5 lm_5^2\} \ddot{\theta}_3] \\
&\quad [+\{(m_4 lm_4^2 + m_5 lm_5^2) s_3 c_3 + (m_4 lm_4 + m_5 lm_5) lm_3 s_3 c_2\} \dot{\theta}_1^2 \\
&\quad (m_4 lm_4 + m_5 lm_5) lm_3 s_{23}\} \dot{\theta}_2^2] \\
&\quad [\{(m_4 lm_4 + m_5 lm_5) c_3\} g]
\end{aligned} \tag{5.3}$$

where the set of terms enclosed in [...] for each joint actuator corresponds to the inertial, centripetal and Coriolis, and gravitational components respectively, while the kinematic and dynamic parameters of the CRS A251 are presented in Table 5.2.

Final torque expressions (5.3) are in hybrid form, in the sense that product terms comprise numerical coefficients (constant kinematic and inertial parameters), as well as symbolic functions of manipulator generalised coordinates and trigonometric functions. General-purpose symbolic algebra tools, such as Mathematica and MAPLE, could have been used for a quantitative significance analysis and symbolic simplification of the general-form dynamic terms, such as those carried out by Corke [12]. This would have been a necessity under real-time constraints for typical 6 DoF robot manipulators, where there are many thousands of terms for each torque expression in closed form. However, the assumptions made in this work led to "relatively" simple robot torque expressions for which simplifications were carried out manually during the derivation, and it was not found necessary to further manipulate the equations.

It is important to be aware of the large computational savings that the above customised dynamics for the CRS A251 have represented with respect to the general closed form dynamics. If Equation (5.2) is expanded for the first 3 DoF of the manipulator, the following matrix structure in general form results:

$$\begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} = \begin{bmatrix} D_{11} & D_{12} & D_{13} \\ D_{21} & D_{22} & D_{23} \\ D_{31} & D_{32} & D_{33} \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \ddot{\theta}_3 \end{bmatrix} + \begin{bmatrix} G_1 \\ G_2 \\ G_3 \end{bmatrix} g$$

[5] This value accounts for the end-effector load, not shown in Figure 5.1, and would also account for the payload, if any, up to the specified maximum of 1 kg.

$$+\begin{bmatrix} H_{111} & H_{112} & H_{113} & H_{121} & H_{122} & H_{123} & H_{131} & H_{132} & H_{133} \\ H_{211} & H_{212} & H_{213} & H_{221} & H_{222} & H_{223} & H_{231} & H_{232} & H_{233} \\ H_{311} & H_{312} & H_{313} & H_{321} & H_{322} & H_{323} & H_{331} & H_{332} & H_{333} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1^2 \\ \dot{\theta}_1\dot{\theta}_2 \\ \dot{\theta}_1\dot{\theta}_3 \\ \dot{\theta}_2\dot{\theta}_1 \\ \dot{\theta}_2^2 \\ \dot{\theta}_2\dot{\theta}_3 \\ \dot{\theta}_3\dot{\theta}_1 \\ \dot{\theta}_3\dot{\theta}_2 \\ \dot{\theta}_3^2 \end{bmatrix} \quad (5.4)$$

which, for the customised dynamics given by Equation (5.3) becomes:

$$\begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} = \begin{bmatrix} D_{11} & 0 & 0 \\ 0 & D_{22} & D_{23} \\ 0 & D_{32} & D_{33} \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \ddot{\theta}_3 \end{bmatrix} + \begin{bmatrix} 0 \\ G_2 \\ G_3 \end{bmatrix} g$$

$$+\begin{bmatrix} 0 & H_{112} & H_{113} & H_{121} & 0 & 0 & H_{131} & 0 & 0 \\ H_{211} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & H_{233} \\ H_{311} & 0 & 0 & 0 & H_{322} & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1^2 \\ \dot{\theta}_1\dot{\theta}_2 \\ \dot{\theta}_1\dot{\theta}_3 \\ \dot{\theta}_2\dot{\theta}_1 \\ \dot{\theta}_2^2 \\ \dot{\theta}_2\dot{\theta}_3 \\ \dot{\theta}_3\dot{\theta}_1 \\ \dot{\theta}_3\dot{\theta}_2 \\ \dot{\theta}_3^2 \end{bmatrix} \quad (5.5)$$

This significant simplification of the closed-form dynamic equations is also due to the particular kinematic design of the manipulator, which has contributed to the elimination of some of the dynamic coupling (D_{ij} and H_{ijk}) between the joint motions. Furthermore, some of the velocity-related dynamic coefficients have only a dummy existence, since they are physically non-existent. In particular, the centrifugal force will not interact with the motion of the joint which has generated it, i.e., $H_{iii} = 0$. It is of interest to notice also the symmetry of the acceleration-related inertia matrix, as well as the partial symmetry of the velocity-related coefficients for each separate joint ($H_{ijk} = H_{ikj}$), general properties of the L-E formulation which, combined, halve the necessary computations in customised form and provide also an initial symbolic validation of the derivation. For a numerical validation of the model, it is important to place the rigid body forces just described within a wider framework, which considers also the contribution of other effects neglected by this model – at least approximately. This is described in Section 5.3.3, yet some notation related to the mechanical transmission system will be first provided that will aid in the understanding of the residual effects.

5.3.2 Motion transmission

The actual components employed in the construction of a robot arm joint usually consist of an actuator (e.g., motor), coupled to the physical joint by a mechanical transmission system. Strictly speaking, the drive train is part of the electro-mechanical subsystem described later in Section 5.4. However, as there is a close affinity between the terminology employed in the following Section, where residual forces are introduced, and the basic definitions of gear trains, it was thought convenient to introduce the fundamental notation first, and not postpone it until that Section.

The transmission is used to direct the actuator motion to the arm joint in order to provide such characteristics as a change of rotational direction, a change of axis, torque multiplication, and speed reduction. The word "transmission" is used to define all the components from the actuator to the physical joint. These components may consist of shaft couplers, belt-and-pulleys, lead screws, etc . Gears, single or as part of a multiple set, are used in almost all industrial manipulators mainly due to the necessity of amplifying the limited torque capability of the majority of industrial motors. The gears amplify the motor torque by a factor N equal to the gear ratio, hence allowing the robot designer to use smaller motors. Referring to Figure 5.4, page 65, an ideal gear train formed by N_m and N_l, $N_l > N_m$, would define a coupling ratio of:

$$N = N_l / N_m \tag{5.6}$$

thus, N turns in the input (motor) shaft would produce a single rotation in the output (link) shaft, i.e.,

$$q_m = N q \tag{5.7}$$

and, provided the torque on the motor shaft is known, the torque on the link shaft can be computed by

$$\tau = \tau_m N \tag{5.8}$$

However, real-world components do not necessarily behave like their ideal models. In fact, they rarely do so. Efficiency η, $0 \leq \eta \leq 1$, is defined as the ratio of the work output to the work input over the same period of time, with the difference being dissipated in friction. This is a dynamic coefficient which is normally assumed constant for simplicity. In this work, a single time-invariant parameter is also employed, as supplied hy the manufacturer. The author is however aware that further investigation into the dynamic properties of the gear efficiency is needed, and that will be highlighted as a further issue of research at the end of Chapter 8. For the ideal gear mechanism of Equation (5.8), the efficiency is 1. However, for the case of a real gear train,

$$\tau = \tau_m N \eta \tag{5.9}$$

This equation reveals that any efficiency less than 1 (100%) will increase the torque required to accelerate a given inertial load. It is important to note that efficiency does not affect the actual transfer ratio of the gears in terms of displacement, velocity, or acceleration, but greatly affects any torque related property.

Gear ratios and efficiencies are collected in Table 5.5, along with the rest of actuator and transmission train characteristics.

5.3.3 Residual dynamic effects

The dynamic equations of motion previously derived in Section 5.3.1 do not encompass all the effects acting on a manipulator. They account only for those structural forces, associated with kinetic and potential energy, which arise from rigid body mechanics. In order to make the dynamic equations reflect the physical device as accurately as possible, it is important to investigate the contribution of a number of residual forces exerted on the arm.

In practice, some of these effects are extremely difficult to model, and even if an approximate model can be found at all, some of its parameters are completely unidentifiable (this is, for instance, the general case for some of the inertial parameters [1]). For a well-designed geared manipulator, the most important source of these forces can be characterised by rotor inertias and friction. The former, obtained here from the technical specifications, can be a very significant characteristic of geared manipulators because a gear ratio of N multiplies the motor's rotor inertia by N^2, as will become readily apparent in the following exposition. The effects of friction can also be modelled as a generalised force applied to the joint of the arm. Friction is a complex non-linear force that is difficult to model accurately, yet in many cases it can have a significant effect on robot arm dynamics. According to An *et al* [1], for the PUMA 600 manipulator at the MIT Artificial Intelligence Laboratory, it was measured that the friction terms accounted for as much as 50% of the motor torques. Craig [9] provided a more conservative estimate of 25% to move a manipulator in typical situations.

Assuming the dynamics of the motor simply described by a rigid load rotating about the shaft axis [6], the equation describing this system is given by (refer to Figure 5.4):

$$\tau_m = J_m \ddot{q}_m \tag{5.10}$$

where J_m is the motor's mass moment of inertia through the axis of rotation (kgm^2), given in Table 5.5 with the remaining actuator's electro-mechanical parameters, and $\ddot{q}_m$ is the angular acceleration of the motor (rad/s^2). When a load is attached to the output gear as in a robot manipulator link, then the

[6] Where any motor friction is considered included in the link friction coefficient.

Joint	$B\ (Nms/rad)$	$F_s(Nm)$
θ_1	0.5	1.0
θ_2	0.8	1.0
θ_3	0.3	0.7

Table 5.3: CRS A251 friction coefficients.

total torque developed at the motor shaft, τ_{total}, is equal to the sum of the torques dissipated by the motor τ_m and its load referred to the motor shaft τ^* as in

$$\tau_{total} = \tau_m + \tau^* \tag{5.11}$$

Where τ^* is, according to Equation (5.9)

$$\tau^* = \frac{\tau + f}{N\eta} \tag{5.12}$$

τ represents the link load torque of each joint as given by the general manipulator equation of motion (5.2), or Equation (5.3) for the CRS A251 in particular. Joint friction is included in $f\ (Nm)$. Although friction might depend on such factors as velocity and position of the joint, to keep the dynamic model simple, its analysis was broken down into velocity dependent viscous friction and starting static friction (or stiction), as given by:

$$f = B\dot{q} \pm F_s \tag{5.13}$$

where B is the viscous friction coefficient (Nms/rad), and F_s represents the starting friction (Nm). Since the contribution of these forces is restricted to a single joint, a simplified decoupled version of model (5.3) for each joint was implemented in MATLAB/SIMULINK. These models where then employed to identify the friction parameters by exploiting the physical insights into the properties of the plant already known, i.e., the dynamic coefficients provided by the manufacturer, to a step input. The magnitude of the "best-fit" coefficients are collected in Table 5.3. The suitability of the parameters and the simplified friction model structure employed is discussed when the overall mechanical model is validated in Section 5.3.4.

Combining Equations (5.2), (5.10), (5.11) and (5.12), the following manipulator equation of motion in matrix form arises:

$$\boldsymbol{J_m\ddot{q}_m} + (\boldsymbol{N\eta})^{-1}\{\boldsymbol{D(q)\ddot{q}} + \boldsymbol{H(q,\dot{q})} + \boldsymbol{G(q)} + \boldsymbol{f(\dot{q})}\} = \boldsymbol{\tau_{total}} \tag{5.14}$$

where $\boldsymbol{J_m}$, $\boldsymbol{N}$, $\boldsymbol{\eta}$ and $\boldsymbol{f}$ are treated here as diagonal matrices for proper operation. Further manipulating this equation, the following expression for the manipulator equations of motion results:

$$\{\boldsymbol{J_m N^2 \eta} + \boldsymbol{D(q)}\}\boldsymbol{\ddot{q}} + \boldsymbol{H(q,\dot{q})} + \boldsymbol{G(q)} + \boldsymbol{f(\dot{q})} = \boldsymbol{\tau_{total} N\eta} \tag{5.15}$$

where the inertial term represents the combined motor plus load reflected inertia, as "seen" by the motor shaft. This is referred to as the "effective inertia matrix", and it is now evident the importance that rotor inertias can play in the overall dynamics of gear manipulators. This explains why some commercial robots are designed with gear ratios that cause rotor inertia to match or dominate link inertias, so that non-linear rigid body dynamics can be neglected altogether in the controller design, hence making control easier [1].

In the following Section, steps are taken to validate this mechanical model, but it is worth noting first that actual manipulator designs are characterised by other side effects which, in some cases, have a significant effect on robot arm dynamics. Some might include:

1. Gear backlash, mainly caused by preloading, tooth wear, misalignment and gear eccentricity, can cause angular displacement of the links. It is extremely difficult to model, but has been assumed greatly minimised in this work by the use of harmonic drives gear trains, and discarded for simplicity.

2. Non-uniform pressure distribution on the contacting surfaces as well as contact deformations of both joints and link surfaces are other factors that can also be taken into account. However, deformations can be assumed to be small enough not to create noticeable changes in the manipulator geometry [5].

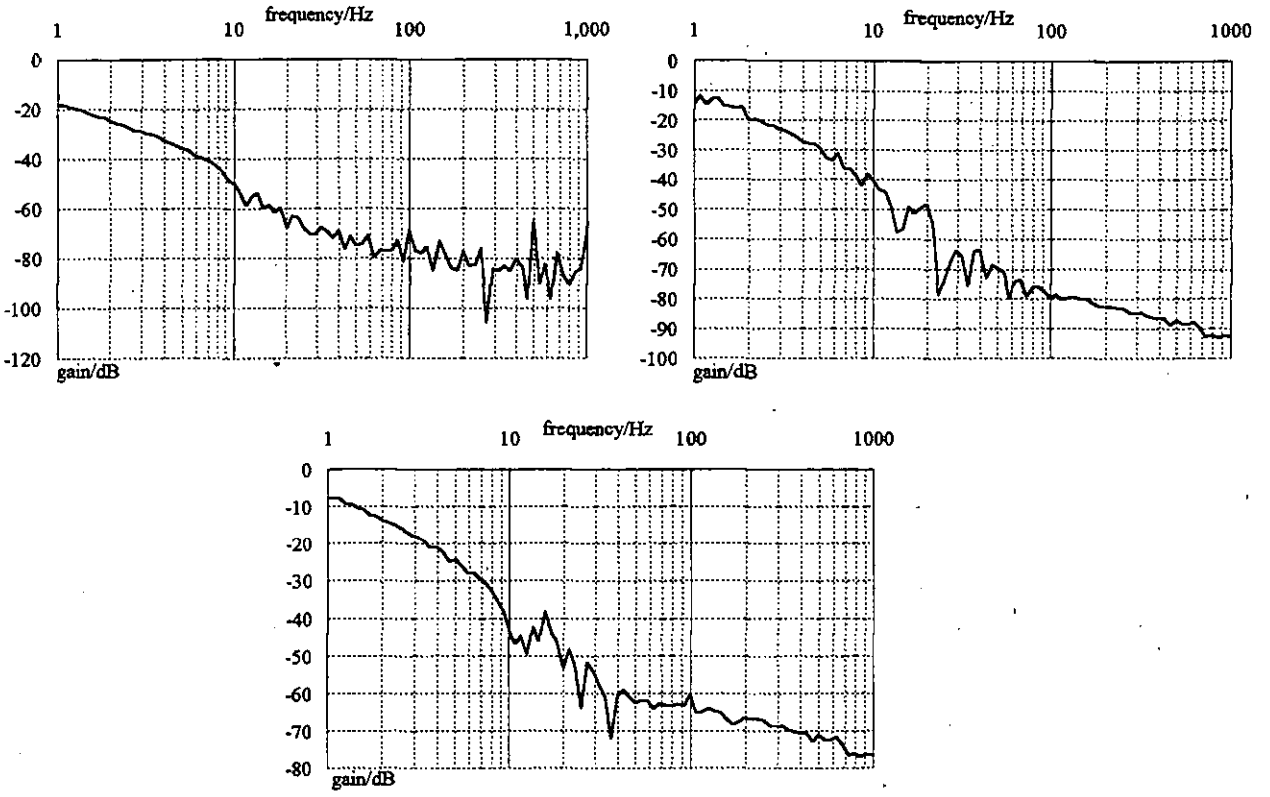

Figure 5.2: Waist (top left), upper-arm (top right) and fore-arm joint frequency response (Hz).

3. Gearing, shafts, bearings and the driven link flexibility. All these elements have finite stiffness, and their flexibility, if modelled, would incur a spring effect that would deflect by varying amounts depending on the load and link configurations. This is particularly true for harmonic drives which have an in-built flexibility due to the flexible-spline. Joint flexibility, if present, will cause loss of accuracy at the end-point, particularly complicating kinematic calibration [1]. The argument for ignoring flexibility effects is that, provided the system is sufficiently stiff, the natural frequencies of these unmodelled resonances are very high, and can be neglected for control system analysis and design, compared to the dominant poles of the system [9]. To validate this assumption, Bode analyses were carried out independently for each joint by means of a Schlumberger Instruments frequency response analyser, the *Solartron 1170*. The procedure involved opening the feedback loop of the uncontrolled system, and applying a range of sinusoidal signals with varying fundamental harmonics and amplitudes. Measurements of the input-output positional gain magnitudes are illustrated in Figure 5.2 for the waist, upper-arm and fore-arm joints. The lack of resonant peaks, in particular in the low frequency range, reveals a stiff frequency response for each manipulator joint. This was an expected result given the rigidity of the light weight construction technique. Since the support of the moving arm – which is part of the waist, consists of a solid cast aluminium foundation for the arm, the response of the waist joint shows a highly stiff characteristic. Furthermore, it can be seen how most of the energy of the system is concentrated within the initial 10 Hz region for all three arm links, hence coinciding with the mechanical resonant frequency of most manipulators, normally around 5 to 10 Hz [6]. The fore-arm and upper-arm joints exhibit a low magnitude higher order effect in the range $15-40$ Hz, which is not present in the waist joint, and has been partially attributed to the stressed aluminium construction of these links, but mainly to the more noticeable physical coupling between these two joints (despite keeping one stationary while exciting the other). In summary, given the low magnitude of the system flexibility, this effect, although always present in any mechanical structure, has been ignored and the arm construction will be assumed that of an ideal rigid body.

5.3.4 Mechanical model validation

Validation of the manipulator mechanical model is the process of examining and assessing the quality and reliability of the model, possibly rejecting its use for the purpose in question. Under the "traditional" system identification approach to building a model, briefly outlined at the beginning of the Chapter, a mathematical function would have been constructed based on a "training" set, in turn obtained from measurements on the real system. Ljung [4] has provided the following reasoning to single out basic validation techniques for the identification process:

1. The most obvious and pragmatic way to decide if a model is good enough, is to test how well it is able to reproduce validation data, i.e., data that were not used to estimate the model, in simulation. The user can then, "by eye" inspection, decide whether the fit is good enough. This is reported by Ljung [4] as the primary validation tool.

2. To determine how far the true system is from the model, that is, to determine model parameter error bounds. If a probabilistic setting is adopted, and the true system is assumed to be found within the chosen structure, it becomes a matter of seeing how much the stochastic disturbances (in essence, the model parameter errors) might have affected the model. Ljung suggests the covariance matrix of the (error) asymptotic distribution as a classical probabilistic measure used for the error bounds [4]. Hence, the predicted model can be assumed to be accurate to within some probabilistic error limits. The fundamental assumption that the true system parameters can be represented in the chosen model structure is normally validated in itself according to 3.

3. To test if the data and the model are consistent with the model structure assumption. This is basically a numerical extension of point 1, where a measurement of the residuals between the model output and the validation data set is obtained. This can be performed "deterministically", where the residual is proved to be always within some bounds, or "probabilistically", where a residual statistical analysis can be performed. For instance, studying whether the residual error signal and the input to the system are independent random variables.

Further insights into the role of model validation for assessing the size of the unmodelled dynamics can be found in another recent paper by Ljung [16]. Although the derivation of the mechanical model (5.15) had not been carried out from observed data, these basic validation features could also be applied to a newly measured input-output validation data set. While other techniques, such as comparing frequency responses of the model and real system in motion, could have been theoretically employed, a combination of point 1 and 3 was regarded as a representative and practical measure of how consistent the model was with the real system. The reason being the frequency analyser available for testing could only measure 1 DoF at a time, thus dramatically limiting the possibility to discern coupling effects between joints. Other techniques include standard software packages for control analysis and design, such as MATLAB/SIMULINK and ACSL. These tools, however, extract a linearised state-space model from the non-linear manipulator model around an operating point, which is then manipulated by linear frequency response routines, such as Bode or Nyquist charts. Given the complexity of the model, where saturations need also to be taken into account for proper operation, the results were, in general, far from satisfactory.

The following procedure was then followed in the analysis:

1. Each joint of the manipulator was made to track a simple cubic polynomial in joint space, as described by Equation (2.3), with polynomial coefficient derived from (2.5), and normalised time $\in [0, 1]$. The configuration data used for estimating the torques was sampled while the manipulator was moving from $(0°, -90°, -45°)$ to $(30°, -57°, -90°)$.

2. A Feedforward controller, Equation (3.8), was implemented to track the desired trajectory.

3. Motion speed, determined by a prespecified time (in seconds) to track the polynomial from the initial to the final point, was varied from fast (0.5), down to medium (0.7) and slow (0.9) motion, thus covering a wide range of speeds and accelerations that could asses the validity of the model within a large configuration spectrum.

4. Measurements of the currents flowing through each motor joint were taken, which according to the motor-torque proportionality relationship, described in the following Section where the actuator

Joint	0.5 s motion	0.7 s motion	0.9 s motion
θ_1	0.965	0.965	0.955
θ_2	0.961	0.98	0.983
θ_3	0.892	0.866	0.862

Table 5.4: Correlation coefficients between model and measured torque at different motion speeds.

model is derived (Equation (5.20)), can be assumed linear with the torque exerted at the motor shaft.

5. The torque was then compared to that obtained from (5.15), and a correlation analysis of both sets of data was also undertaken as a residual analysis test. This coefficient, calculated across measurement data tests is obtained as follows:

$$\frac{\sum(d_i - \bar{d})(o_i - \bar{o})}{\sqrt{\sum(d_i - \bar{d})^2 \sum(o_i - \bar{o})^2}} \qquad \bar{d} = \frac{1}{N}\sum_i d_i \qquad \bar{o} = \frac{1}{N}\sum_i o_i \tag{5.16}$$

where N= measurement data set size, d_i represents the torque output calculated from the model, o_i is the actual measured torque, and the summations are over the measurement data set. This is a simple, normalised overall measure of how well desired and actual outputs correlate:

$$\begin{aligned} 1 &\Rightarrow \text{ perfectly correlated} \\ 0 &\Rightarrow \text{ completely uncorrelated} \\ -1 &\Rightarrow \text{ perfectly uncorrelated} \end{aligned}$$

The CRS A251 robot arm has an optical encoder on each of its five joints for the feedback of positional information, but lacks tachometers. Therefore, it was necessary to numerically differentiate the positions and resulting velocities to obtain the robot configurations. Positions were sampled at 250 Hz, while velocity and acceleration were filtered using a digital low-pass filter with cut-off frequencies of 28 Hz and 4 Hz respectively, determined empirically to provide the best (smoothest) results. Because the chosen cubic polynomial is discontinuous on the acceleration (see Section 2.4), a low cut-off frequency had to be chosen in the experiments to avoid exciting unmodelled dynamics. The resulting smooth configurations for each joint along the three proposed trajectories are not substantially relevant and thus are presented in Appendix D, for completeness. A non-intrusive Hall effect current transducer was employed to measure the torques. This is a fast response current sensor [7], having virtually no effect on the circuit loading. Figure 5.3 shows a comparison of the measured torque and the computed torque generated by the manipulator model for the three joints at the different speeds, while the corresponding correlation coefficients for these sets of data are collected in Table 5.4. The three sets of figures for each joint match fairly well, hence verifying qualitatively the accuracy of the model, which is further validated by the quantitative residuals analysis. This suggests that, for the purposes of this work, even a poor approximation of the mass moment of inertia parameters will allow good estimates of the total torque necessary to achieve a desired trajectory.

5.3.4.1 Sources of error

A model structure is always too simple to fully describe a real system. Sources of error are numerous and their importance should, at least, be considered:

1. The ultimate source of error is the random noise inherent in the sensing process itself. The noise level on the position sensing was negligible, with a maximum [8] of around $\pm 4.6\%$, and was not increased by the digital circuitry designed to interface the industrial controller with the measurement/controller PC. The design is described in further detail in the next Chapter as part of the controller implementation.

[7] Response time $< 1\mu$sec which can accurately follow signals of > 50 A/μs, exhibiting a frequency range of up to 100 kHz.

[8] This was measured over a fitted smooth moving average curve of period 3.

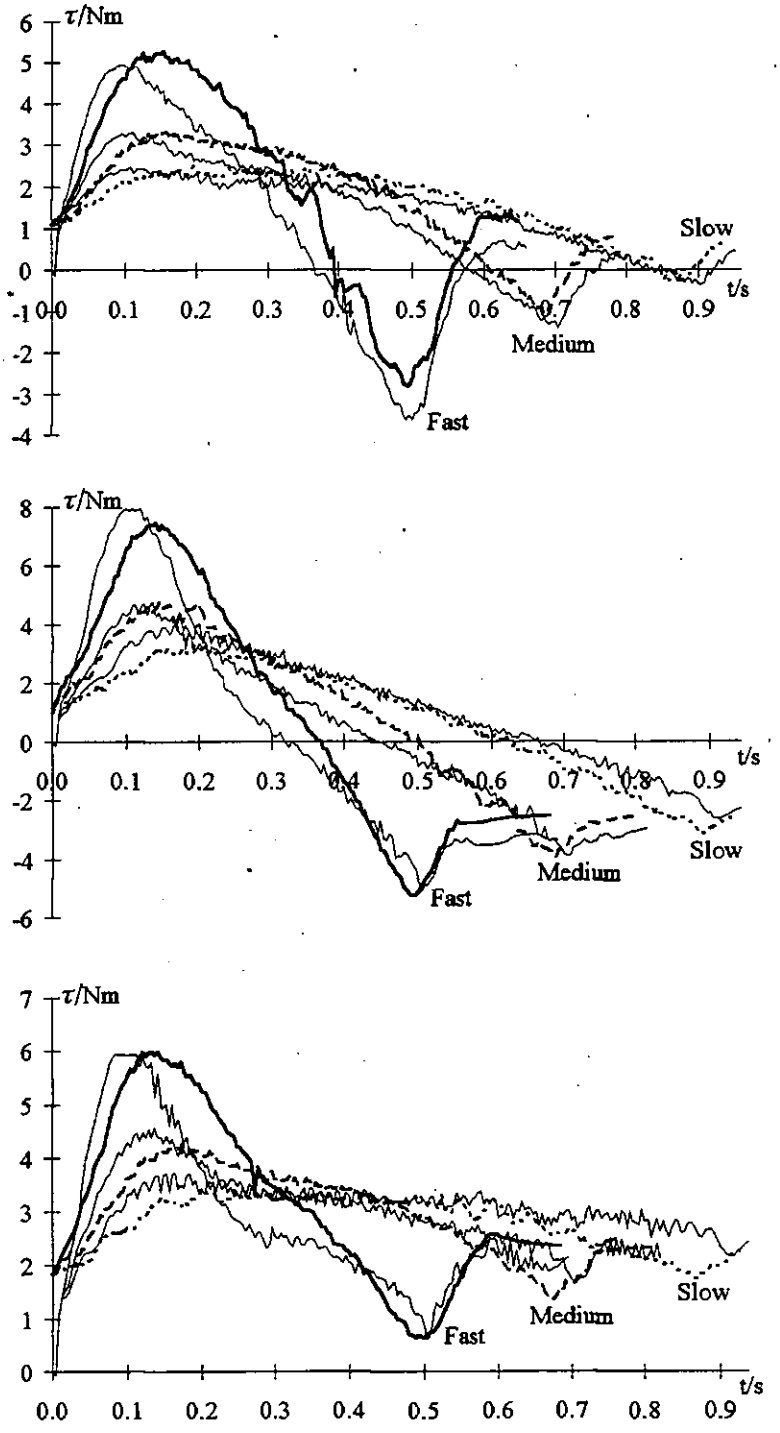

Figure 5.3: Waist (top), upper-arm (middle) and fore-arm measured (thin curve) and computed (bold curve) torque responses, for tracking fast, medium and slow polynomial trajectories.

2. In addition to unavoidable random noises, there might be more unmeasurable input signals that could affect the output. These signal sources that can not be traced are called "disturbances", and we simply have to live with the fact that they will have an adverse effect on the comparisons.

3. A further source of noise is unmodelled dynamics, or "bias error" [4]. It has already been pointed out that robot links are not perfectly rigid bodies. However, the structural analysis of the CRS A251 showed that compliant effects could be safely overlooked. Moreover, the contribution that non-linear frictional effects might have had in the torque balance have been neglected in the linear frictional model (5.13).

4. Torque in an armature-controlled DC motor is, theoretically, a linear function of the armature current, and correspondingly, the armature voltage, as will be described in the following Section. However, due to bearing friction at low torques and saturation characteristics at high torques, the actual current(voltage)-torque curves are not linear [6]. For these reasons, motor current, as obtained from Equation (5.20), can only but approximate joint torques. The solution at hand is a conversion of computed torque to required input current (voltage), which can be accomplished via lookup tables or calculations from piecewise linear approximation formulae. This alternative, though, is problematic and time-consuming to set up, thus seldom employed, while the linear motor current-torque relationship is the widely adopted approach, despite the possible errors.

5. Finally, the need to (twice) differentiate the position numerically to find velocity and acceleration, greatly amplifies whatever noise is present. Differentiation of a signal always decreases the signal-to-noise ratio because noise generally fluctuates more rapidly than the command signal, and should be avoided when possible [17]. Two methods are readily available to estimate these unmeasured state variables without an explicit differentiation process:

   a. Integrating the manipulator equations of motion, given by (5.15). This derivation would only be required once in order to obtain the velocity, since the acceleration could then be calculated by simply rearranging Equation (5.15). An example of this method can be found in An *et al* [1].

   b. Design a *state observer* that could estimate, or observe, the state variables based on the measurement of the output and control variables. A few examples of this method can be found in Ogata [17].

   Both methods would take the manipulator model into account in the first place, hence introducing a good deal of extra computation in the loop. Moreover, in reference to the first method, An *et al* point out that since an integrator is an infinite gain filter at zero frequency, large errors can result from small low frequency errors, such as offsets. Therefore, in their experiments, the estimated results were not as good as expected. In view of these facts, and in order to make the on-line implementation more straightforward, a numerical differentiation was the preferred solution, coupled with low-pass filters as described earlier on page 62, to limit the effect of the introduced noise.

5.4 Electro-mechanical Modelling

The previous Section has described how the motor torques required to drive the manipulator arm can be obtained from known physical laws such as the Lagrangian formulation of mechanics. The control law expression, Equation (5.15), computes the mechanical torque set point that serves as the input to the robot arm actuators. However, the inability of commercial robots to control joint torques is a well known problem [1, 18]. Commercial motor servos are typically position controllers, to which one can only send positional set-points (see Section 3.4). Yet virtually all advanced control strategies are designed on the capability of controlling joint torques, as shown by the majority of control strategies presented in Chapter 3, including the optimal control strategy developed in this work. Having computed a nominal command torque signal for some specific motion, two alternatives are at hand to implement it on an industrial manipulator:

1. It can be integrated, as will be described in Section 5.5.1, using the derived model of the rigid body dynamics, to produce the joint positions, which can then be used as a reference trajectory

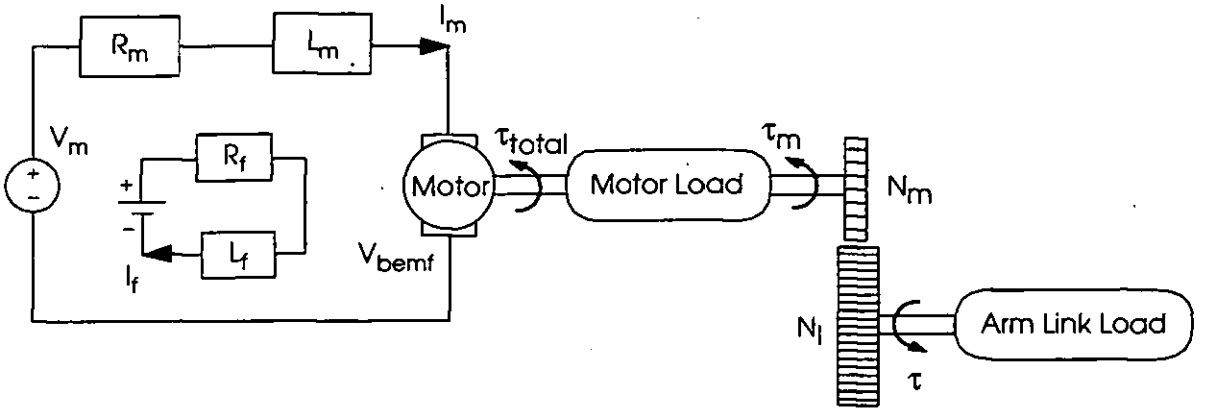

Figure 5.4: Schematic diagram of permanent magnet DC servomotor, gear train and link load.

to the position controllers fitted in the manipulators. An example of such an approach, where an optimal control trajectory is computed to track a specified end-effector path, can be found in Shiller [19]. Ignoring motor and driver dynamics are the main reasons noted by the author to explain the noticeable bias between nominal and actual torque and trajectory.

2. To directly control the motor currents, so that the torque can be employed as the command signal. Although this approach can, theoretically, compensate for the transient actuator dynamics, it also poses an implementation problem, since it becomes then necessary to **reverse engineer** the motor structure and drive amplifier to derive the physical driving signal. This is often a difficult problem, related to the unwillingness of robot manufacturers to provide specifications to assist in this endeavour for proprietary and safety issues.

Apart from the obvious advantage of the second alternative to compensate for the dynamics of the underlying physical actuator system, there are two additional practical issues which explain why this was the preferred solution for the work presented here:

1. For the derivation of an optimal controller, the torque profile should first be computed in order to obtain the optimal input trajectory to the positional servos. An exact numerical method should then be employed to solve the TPBV problem, as described in Chapters 3 and 4, which would prevent the solution to be implemented on-line.

2. In order to meet the demands of a real-time application, the derivation of the control signal as the output of a SISO system (single actuator/driver) is also faster to compute than the forward integration in time of the mechanical manipulator rigid body MIMO model required by the first approach.

Therefore, in this Section, the dynamic characteristics of the actuators actually driving the joints of the robot arm shall be modelled. These will be employed to convert the computed mechanical control torque into the applied motor control signal, as just mentioned, as well as to conduct accurate simulations.

Industrial manipulators are generally driven by either DC, AC or stepper electro-motors, hydraulic, or pneumatic actuators. Robots with a closed-loop control system are usually driven by permanent magnet DC electro-motors, as is the case of the CRS A251 actuators, driven by EG & G Torque Systems M2110 series motors. Thus, in the following, the modelling of these actuators shall be considered. However, these considerations might be easily extended to hydraulic actuators [20].

5.4.1 Permanent magnet DC servomotor

Although motor models can be quite complicated, they are in some sense simpler than rigid link dynamic models, because motor dynamics are typically confined to a single joint. This reduces motor modelling to a SISO problem, rather than the more difficult MIMO problem. Motor models generally

Waist Electro-mechanical Subsystem (θ_1)			
Parameter	Description	Value	Unit (S.I.)
J_m	Motor Inertia [9]	2.81E-05	kgm^2
K_m	Motor Torque Constant	0.0657	Nm/A
K_{bemf}	Motor back e.m.f. Constant	0.0657	Vs/rad
T_m	Motor Time Constant	1.9E-0.3	s
R_m	Motor Resistance [10]	2.32	Ω
L_m	Motor Inductance [11]	4.41E-0.3	H
N	Gear Ratio	72	
η	Gear Efficiency	0.75	

Upper-arm Electro-mechanical Subsystem (θ_2)			
Parameter	Symbol	Value	Unit (S.I.)
J_m	Motor Inertia [9]	2.81E-05	kgm^2
K_m	Motor Torque Constant	0.0657	Nm/A
K_{bemf}	Motor back e.m.f. Constant	0.0657	Vs/rad
T_m	Motor Time Constant	1.9E-0.3	s
R_m	Motor Resistance [10]	2.32	Ω
L_m	Motor Inductance [11]	4.41E-0.3	H
N	Gear Ratio	72	
η	Gear Efficiency	0.75	

Fore-arm Electro-mechanical Subsystem (θ_3)			
Parameter	Symbol	Value	Unit (S.I.)
J_m	Motor Inertia [9]	2.81E-05	kgm^2
K_m	Motor Torque Constant	0.0657	Nm/A
K_{bemf}	Motor back e.m.f. Constant	0.0657	Vs/rad
T_m	Motor Time Constant	1.9E-0.3	s
R_m	Motor Resistance [10]	2.32	Ω
L_m	Motor Inductance [11]	4.41E-0.3	H
N	Gear Ratio	72	
η	Gear Efficiency	0.56	

Table 5.5: Actuator and gear train electro-mechanical characteristics.

include not only the structure of the motor and amplifier, but also properties of the drive train. The reader is referred to previous Section 5.3.2, where the mechanical transmission gearing was intentionally presented as part of the mechanical arm subsystem for convenience in the flow of the presentation.

In essence, a permanent magnet DC motor is an armature excited, continuous rotation actuator incorporating such features as high torque-power ratios, smooth, low speeds operation, linear torque-speed characteristics, and short time constants. Use of a permanent magnet field and DC power provides maximum torque with minimum input power and minimum weight [6]. A schematic linear circuit model of a permanent magnet armature-controlled DC servomotor, gearbox and mechanical load is shown in Figure 5.4.

The electro-mechanical characteristics for the actuator driving each joint are collected in Table 5.5. To develop a dynamic model for this actuator, Kirchhoff's voltage law was applied around the armature windings which yields:

$$V_m = L_m \dot{I}_m + R_m I_m + V_{bemf} \tag{5.17}$$

[9] This value accounts also for the Harmonic Drive's wave generator inertia of 3.27E-0.6 kgm^2.

[10] This value was measured to around 4.6Ω at the point where the control signal is injected into the motor. This is probably so because in addition to the provided armature resistance, terminal resistance and that of the armature magnetic losses increases this value. Also, because of its very low magnitude, contributions made by wiring should also be considered.

[11] Obtained according to the inductive circuit relationship $L_m = T_m R_m$.

V_{bemf} is called the back electromotive force voltage, and is an internal voltage that counteracts V_m and is produced when the armature rotates in a DC magnetic field. This voltage builds up linearly as the motor shaft angular speed $\dot{q}_m$ increases, that is:

$$V_{bemf} = K_{bemf}\dot{q}_m \tag{5.18}$$

where K_{bemf} (Vs/rad) is referred to as the back e.m.f. constant of the motor.

Having derived the mechanical characteristics of motor, gears and link for each joint, the relationship between the electrical and mechanical components of the system needs to be established in order to relate the control torque action to the underlying physical control variables which actually excite the actuator. The electrical and mechanical subsystems are coupled to one another through an algebraic torque equation. In general, the torque developed at the motor shaft is proportional to the product of two currents, the armature winding current I_m and the field winding current I_f through the air gap flux ψ :

$$\tau_{total} = \psi K_1 I_m = K_f I_f K_1 I_m \tag{5.19}$$

Where K_f is the flux constant, and K_1 is also a constant. However, in an armature-controlled DC motor, the field winding current is constant, so that the flux also becomes constant. Consequently, the torque developed at the motor shaft is assumed to increase linearly with the armature current, independent of speed and angular position, according to:

$$\tau_{total} = K_m I_m \tag{5.20}$$

where K_m is called the motor-torque constant (Nm/A) which provides the required relationship. As seen in Table 5.5, K_m and K_{bemf} correspond to the same physical constant under compatible units.

5.4.2 Amplifier stage

A servo amplifier must be used to convert the low-power command signal that comes from the controller to levels that can be used to drive the joint motor. In general, two drive source configurations are widely available: pulse width modulated (PWM) and linear amplifiers. The latter, in turn, can incorporate either voltage or current feedback (or both). Despite the more cost-effective solution provided by the PWM approach, where the power dissipation is only a fraction of that with linear amplifiers, the CRS A251 is fitted with voltage (or velocity) linear amplifiers as the analogue driving source. This is a disadvantage with respect to the simpler (from the control point of view) current amplifiers, since the issue of deriving the analogue control signal from the commanded torque must take into consideration the full electrical behaviour of the motor, as given by Equation (5.17), whereas only the motor torque constant would have been necessary to model the entire motor dynamics in current-driven DC motor configuration [12], with the consequent reduction of possible error sources.

The overall block diagram for a single joint, obtained from Equations (5.15), (5.17) and (5.20), combined with the linear voltage amplifier, can then be depicted as shown in Figure 5.5, where K_{amp} is the gain of the amplifier, calibrated with a voltage gain stage of 2 ($\pm 10\%$) for all the joints, and V_{in} represents the voltage command signal from the controller.

5.4.3 Electro-mechanical model validation

The same methodology employed in Section 5.3.4 to validate the mechanical model was also applied here to assess the validity of the motor/driver model. In this case the "inverted" electro-mechanical model was studied to compute how closely it could derive the voltage command to the driver as a function of the desired nominal output torque applied at the motor shaft. Or, in other words, how adequate was the model to "emulate" the operation of a current-driven motor, so that the motor/driver could be commanded in torque mode.

As before, the nominal torque command was computed from a Feedforward controller, implemented to make the manipulator follow a cubic trajectory from initial to final point. Since the actuator model is not as dependent on configuration as the full rigid body dynamics, only the average speed polynomial (i.e., that taking 0.7 seconds to be completed, as described in Section 5.3.4) was employed as the

[12] This type of amplifier gives a constant output current for a given input voltage.

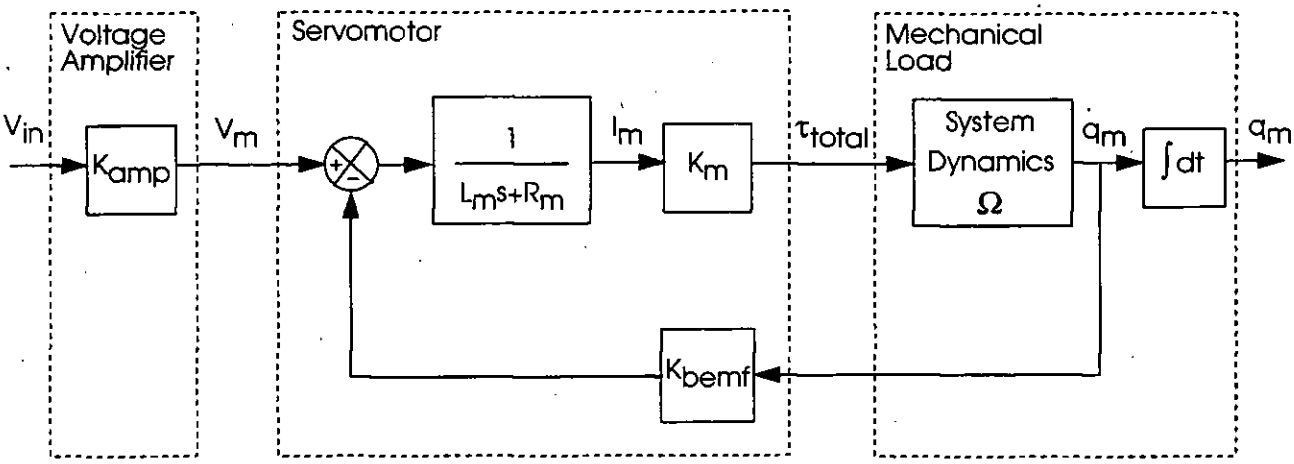

Figure 5.5: Block diagram of electro-mechanical system and mechanical load.

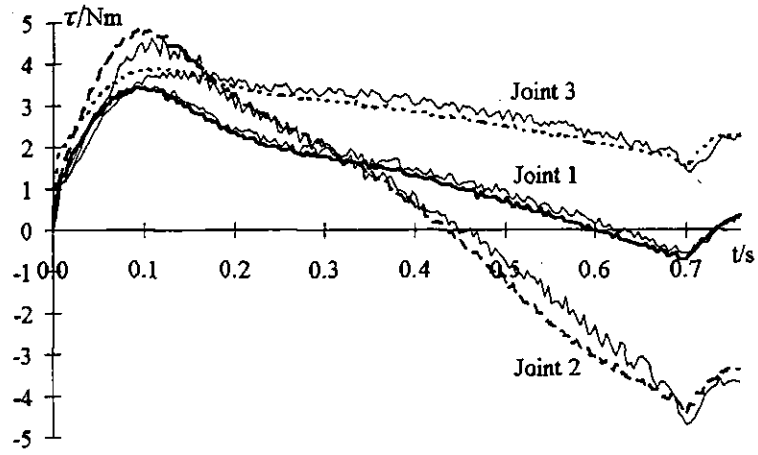

Figure 5.6: Waist (Joint 1), upper-arm (Joint 2) and fore-arm commanded torque (bold curve) and measured torque response from actuator model, for tracking medium speed polynomial trajectory.

reference trajectory [13]. This demand input torque, τ_{total}, denoted in bold in Figure 5.6, was then applied to the inverted joint actuator model, derived by equating (5.17) and (5.20) to eliminate I_m and obtain the corresponding control voltage, given by:

$$V_m = \frac{1}{K_m}(R_m \tau_{total} + L_m \dot{\tau}_{total}) + K_{bemf}\dot{q}_m \tag{5.21}$$

The measured torque response, depicted by the thin curve in Figure 5.6, corresponds to the actual actuator torque, sensed again by the Hall effect transducer, which resulted from applying the commanding voltage signal obtained from Equation (5.21) to the motor driver.

Despite overlooking some unmodelled actuator dynamics, commanded torques are shown to be in close agreement with the measured actuator model torques, as the correlation coefficients in Table 5.6 also prove. The majority of the error sources found to affect the accuracy of the rigid body dynamics in Section 5.3.4.1 are also applicable to the electro-mechanical subsystem, and will not be repeated here. Additional non-linear effects not considered in the motor/driver model might include deadzone for small torques, cogging, and imperfect commutation circuitry and consequent position measurement errors. Special mention should be given to the magnitude of the armature resistance, a complex and difficult parameter to model which is notoriously prone to drift under the effects of temperature variations. In order to minimise this effect during the experiments presented in this Chapter, data were taken after the controller had been left on for a while to warm up, thus providing consistent results in successive runs.

[13] Faster motions are also shown and discussed in Chapter 8, when the electro-mechanical model is employed with the proposed optimal controller.

Joint Actuator	Correlation Coefficient
θ_1	0.993
θ_2	0.992
θ_3	0.922

Table 5.6: Correlation coefficients between commanded torque and measured model torque.

Other alternatives to account for the motor/driver dynamics have been proposed. For instance, Shiller *et al* [18] developed a simplified empirical linear model, called the "viscous friction model", as a substitution for the comprehensive motor model, which was shown valid for the speeds encountered in their experiments. An *et al* [1], on the other hand, proposed the reduction of the motor non-linear effects by implementing an additional torque feedback loop at each joint. For the purpose of the work undertaken in this project, the validation results presented suggest that torques could he commanded accurately at each joint by deriving the voltage command to the driver according to Equation (5.21).

5.5 Dynamic Simulation

The dynamic models developed in this Chapter were employed not only in the design of the optimal controller, described in the following Chapter, but also to simulate the behaviour of the robot manipulator. The purpose of systems study through modelling is to aid the analysis, understanding, design, operation, prediction and/or control of systems without actually constructing and operating the real process [21]. Models play the role of the real objects whose analysis by real experimentation could be expensive, risky, time-consuming or even physically impossible [22]. Simulation models have traditionally been approached by textual-based computer simulation languages, both discrete (GPSS, SIMULA, etc.) and continuous (ACSL, CSMP, etc.) some of which provide, at most, the capability to plot some simulation results in a simple graphical environment. However the rapid development of computer hardware and graphics software during the last decade has added a new dimension to the practice of modelling and simulation. In this work, traditional numerical simulation techniques have been coupled with advanced graphics to get the most out of the simulation process.

5.5.1 Numerical simulation

When dynamics are to be computed for the purpose of performing a numerical simulation of a robot manipulator, the issue reduces to solving the arm dynamics model, Equation (5.15), for the current acceleration of the manipulator link, given current state $(q, \dot{q})$ of each link of the robot manipulator, and commanded control action τ. This, in turn, can then be integrated numerically over the simulation interval Δt to compute future position and velocities. In this work, two of the best well-known algorithms were implemented, Euler and multi-step Runge-Kutta order four [23], the former being only used for fast simulations due to its simplicity and limited accuracy.

5.5.2 Graphical simulation

The novel environment of advanced computer graphics in control design was explored to simulate the response of the new controller. It is generally accepted that humans can relatively easily assimilate complex information from pictorial images. As *Confucius* once said,

> *"A picture says more than a thousand words."*

Undoubtedly, colour graphics and animation are considered a highly desirable feature in understanding the dynamics of system behaviour via simulation software. Indeed, this is found particularly attractive in robotics where graphical programming has emerged as the natural way to plan complex robot motions safely, quickly, and easily [24].

For the development of the new motion strategy described in the next Chapter, a suite of tools and technologies capable of matching the capabilities of the human user to the requirements demanded by the application was sought. Deneb's graphical robot simulation software, TELEGRIP^{TM} [25, 26], provided the virtual reality environment required (see Appendix E). Desktop virtual reality is an

advanced concept for graphical design, prototyping and systems simulation which makes the designed objects' bebaviour more accessible and understandable to the user. The attributes and associations between objects in a virtual environment permit an approximation to the nature and bebaviour of such objects and/or processes which do not yet exist, thus providing the sort of front-end with which the user feels comfortable and accelerating the overall testing and development process.

The design process takes place in three stages:

1. A solid object is first represented in a CAD package using primitive solids such as cubes, cones, wedges, spheres, etc. which are added, subtracted, cut, etc. to form desired shapes for the robot parts and its operating environment.

2. These are then fed into the graphical simulation package where further non-geometric attributes such as motion definition, joint limits and speeds, input/output, dynamic characteristics, etc. are attached to the solid model of the manipulator and devices in its surroundings.

3. Traditional numerical simulation algorithms, such as those mentioned in the previous Section 5.5.1, can then be linked to the software to visualise the dynamic response of the system.

The stand-alone CAD solid model of the CRS A251 industrial robot used for the controller simulations in this work is shown in Figure 5.1. This was employed, along with a number of other workstations, to simulate and improve the automated radiopharmaceutical dispenser mentioned at the beginning of this thesis as the main motivation behind this work. Although an exciting and new area of development, this dissertation would go off the main track by getting into more detail about advanced computer graphics and control. Hence, the reader is referred to Miró *et al* [27] for more details on the actual graphical simulation implementation.

5.6 Summary and Discussion

The work reported in this Chapter has presented the derivation of the dynamic model of the CRS A251 industrial robot manipulator from known physical laws and relationships. This was not limited to the body structure dynamics, which experimental results showed appropriately approximated by Lagrangian rigid body mechanics, but also to the often ignored motor and driver dynamics, which were reverse engineered to be able to drive the actuators in torque mode. To decide whether the model and the data were indeed consistent with the assumptions made about the model structure, a visual comparison coupled with a quantitative residual error analysis demonstrated that the predicted output could substantially account for the most significant plant dynamics. This work, though, might be regarded as preliminary in that an adequate statistical characterisation of the errors between measured and predicted torques has not been attempted. Nevertheless, some insights were gained into the sources of such errors.

While experimental results proved the adequacy of the model for joints 1 and 2, the unmodelled dynamics of the lighter third link, including motor dynamics and residual friction, were shown to be dominant and yielded larger torque errors than the other two joints. Detailed inspection of the robot revealed that a preloaded roller chain was used to transmit the rotational motion to the joint link, thus allowing the actuator to be positioned closer to the base. The result is a design with lower inertia associated with the moving parts, thus allowing faster robot motions. Additional positive effects included increased stiffness, the virtual elimination of backlash, and the enhancing of rotational accuracy because of more uniform loading of the rolling bodies. However, some unmodelled negative effects can also be associated with the gear chain, such as higher loading of the bearing components, possibly affecting friction and, associated with it, higher working temperature and energy losses [5]. Although these effects were not included in the model, the overall successful match of model and measured data suggested that, for control purposes, the proposed model could provide good estimates of the real system, with correlation coefficients $\in [0.86, 0.99]$ depending on the joint and speed of motion.

References

[1] An C.H., Atkeson C.G., and Hollerbach J.M. *Model-Based Control of a Robot Manipulator.* MIT Press, (1988).

[2] Phillips C.L. and Harbor R.D. *Feedbock Control Systems.* Prentice-Hall Inc., third edition, (1996).

[3] Armstrong B., Khatib O., and Burdick J. The explicit dynamic model and inertial parameters of tbe puma 560 arm. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 510–518, San Francisco, USA, April (1986).

[4] Ljung L. From data to model: a guided tour. In *Proceedings of IEE CONTROL'94*, pages 422–430, March (1994).

[5] Rivin E.I. *Mechanical Design of Robots.* McGraw-Hill Book Co., (1988).

[6] Fu K.S., Gonzalez R.C., and Lee C.S.G. *Robotics: control, sensing, vision, and intelligence.* McGraw-Hill Book Co., (1987).

[7] Hollerbach J.M. A recursive formulation of lagrangian manipulator dynamics. *IEEE Transactions on Systems, Man and Cybernetics*, **SMC-10**(11):730–736, (1980).

[8] Li C.J. A new lagrangian formulation of dynamics for robot manipulators. *Journal of Dynamic Systems, Measurement, and Control. Transactions of the ASME*, **111**:559–567, December (1989).

[9] Craig J.J. *Introduction to Robotics: mechanics and control.* Addison-Wesley Publishing Company Inc., second edition, (1989).

[10] Brady M., Hollerbach J.M., Johnson T.L., Lozano-Perez T., and Mason M.T. *Robot Motion: planning and control.* MIT Press, (1982).

[11] Kane T.R. and Levinson D.A. The use of kane's dynamical equations in robotics. *International Journal of Robotics Research*, **2**(3):3–20, Fall (1983).

[12] Corke P.I. An automated symbolic and numeric procedure for manipulator rigid-body dynamic significance analysis and simplification. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1018–1023, April (1996).

[13] Sanders D.A. *Making Complex Machinery Move - automatic programming and motion planning.* Robotics and Mechatronics Series. Research Studies Press Ltd., (1993).

[14] Abdalla E., Pu H.J., Müller M., Tantawy A.A., Abdelatif L., and Eldin H.N. A novel parallel recursive newton-euler algorithm for modelling and computation of robot dynamics. *Mathematics and Computers in Simulation*, **37**(2-3):227–240, (1994).

[15] Mignon F. Implementation of a new parallel recursive newton-euler algorithm for robot dynamics in a net of transputers. Master's thesis, School of Mechanical and Manufacturing Engineering, Middlesex University, UK, (1995).

[16] Ljung L. and Guo L. The role of model validation for assessing the size of the unmodeled dynamics. *IEEE Transactions on Automatic Control*, **42**(9):1230–1239, September (1997).

[17] Ogata K. *Modern Control Engineering.* Prentice-Hall Inc., second edition, (1990).

[18] Shiller Z., Chang H., and Wong V. The practical implementation of time-optimal control for robotic manipulators. *Journal of Robotics & Computer-Integrated Manufacturing*, **12**(1):29–39, (1996).

[19] Shiller Z. Time-energy optimal control of articulated systems with geometric path constraints. *Journal of Dynamic Systems, Measurement, and Control. Transactions of the ASME*, **118**:139–143, March (1996).

[20] Klafter R.D., Chmielewski T.A., and Negin M. *Robotic Engineering. An Integrated Approach.* Prentice-Hall Inc., (1989).

[21] Neelamkavil F. *Computer Simulation and Modelling.* John Wiley & Sons Ltd., (1987).

[22] Matko D., Karba R., and Zupančič B. *Simulation and Modelling of Continuous Systems. A case study approach.* Prentice Hall Int.(UK) Ltd., (1992).

[23] Burden R.L. and Faires J.D. *Numerical Analysis*. PWS-KENT Publishing Company, fourth edition, (1989).

[24] Smith L.A. and Barnes S.A. The use of computer graphics to assist real-time manipulator operation. In *Proceedings of the European Robotics and Intelligent Systems Conference - Euriscon'94*, volume 2, pages 637–646, Malaga, Spain, (1994).

[25] Deneb Robotics Inc. *TELEGRIP Monuol*, 3.0 edition, March (1996).

[26] Dionise J.A. On the dynamic simulation of robot manipulators using hamiltonian equations of motion. In *Deneb User Group 1993 Proceedings*, pages 47–48. Deneb Robotics, Inc., (1993).

[27] Miró J.V., Stoker M., and Gill R. The use of computer graphics for robot motion simulation. In *Proceedings of the 11th ISPE/IEE/IFAC International Conference on CAD/CAM, Robotics & Factories of the Future*, pages 884–889, Pereira, Colombia, August (1995).

Chapter 6

Controller Design and Analysis

6.1 Introduction

The different aspects of the core research work developed in previous Chapters are employed in this Chapter to describe a practical example showing how optimal control theory can be applied to the problem of unconstrained point-to-point manipulator trajectory planning and control.

The preference in previous Chapters was to examine the theoretical developments behind the various optimal strategies, with some explanatory remarks. This Chapter, on the other hand, is devoted to the practical analysis of Pontryagin's Maximum Principle (MP) applied to the design of a controller for an industrial manipulator. This problem is first represented in the phase-plane space in Section 6.2 because, as will become readily apparent during the remainder of the Chapter, the controller is implicitly designed by means of the phase-plane technique. A key remark about the variability of maximum actuator bounds is given in Section 6.2.1.

In Section 6.3, the formulation of the open terminal-time control problem, in which the objective is to transfer the system from an arbitrary initial state to a specified target set in minimum time, is briefly placed in context within the general framework of classical optimal control problems. Before tackling the complex problem of designing an optimal controller for a robotic manipulator, a detailed exposition of the double integrator problem is given in Section 6.4. The problem might be regarded as the simplest possible non-trivial manipulator system, where the model is a simple inertial mass, yet it displays many of the important theoretical peculiarities of the general class of optimal control problems solved via the MP. This is then followed by Section 6.5, where a complete treatment of the design of an optimal controller for the coupled non-linear manipulator plant, whose dynamic equations of motion were developed in the preceding Chapter, is given. It is shown there how, under certain assumptions, the problem can be treated as a quasi-double integrator problem, whose dynamics need updating at each sample interval. While these conjectures render the controller as a "near-optimal" strategy, they nevertheless allowed an analytical solution which is feasible to be implemented on-line, as shown in the following Chapter.

The full structure of the controller is then presented in Section 6.5.1, while the implications of the hypothesis undertaken, along with the significance of other issues such as its feedback form or piecewise linear dynamics, are extensively studied in Section 6.5.2.1. Finally, an analysis of the stability of the controller along the derived trajectory is presented in Section 6.5.3. The final remarks drawn in Section 6.6 about the design of the near-optimal trajectory planner/controller with the proposed strategy conclude the Chapter.

6.2 Phase-Plane System State Representation

From the equations of motion of the plant, derived in Section 5.3.1 and 5.3.3, it was shown that the robot manipulator could be described by a non-linear second order multi-variable expression, as given by Equation 5.15. As pointed out in Section 3.3.1, state-space constitutes the fundamental representation for optimal control theory, thus the analysis of the plant should commence by appropriately selecting the state variables to be employed. In this work, system variables were chosen equal to the phase variables, namely the system output and its first derivative (see definition 3.3), both for simplicity in

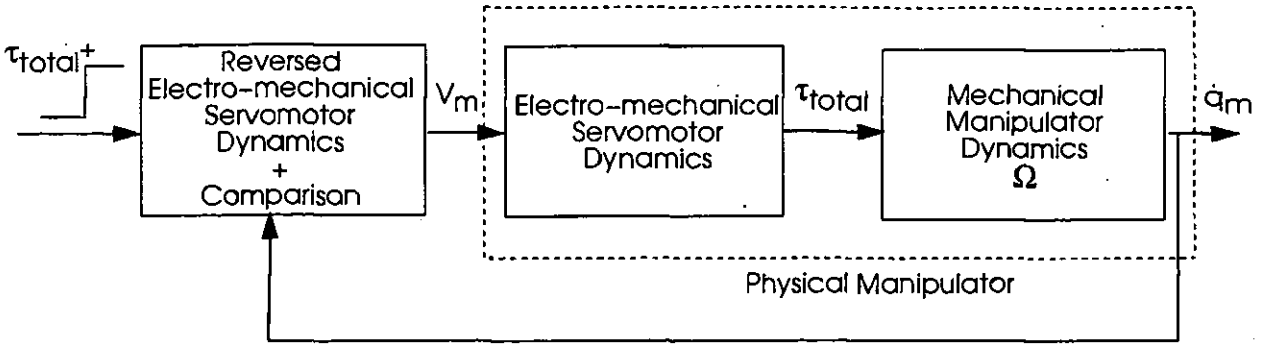

Figure 6.1: Experimental setup for the verification of admissible controls.

the formulation and synthesis of the controller, and because the "Phase-Plane" analysis tools [1, 2] for the study of non-linear systems will feature strongly in the remainder of this Chapter.

Hence, the following $2n$-dimensional state vector will be used to rewrite the dynamic equations:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x_1} \\ \boldsymbol{x_2} \end{bmatrix} = \begin{bmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} \tag{6.1}$$

With this state representation, Equation 5.15 can be rewritten in the following state-space form:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}(\boldsymbol{x}) + \boldsymbol{B}(\boldsymbol{x})\boldsymbol{u} \tag{6.2}$$

where

$$\boldsymbol{A}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{x_2} \\ -\boldsymbol{D_{eff}^{-1}}(\boldsymbol{x_1})[\boldsymbol{H}(\boldsymbol{x_1}, \boldsymbol{x_2}) + \boldsymbol{G}(\boldsymbol{x_1}) + \boldsymbol{f}(\boldsymbol{x_2})] \end{bmatrix} \quad \boldsymbol{B}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{D_{eff}^{-1}}(\boldsymbol{x_1}) \end{bmatrix} \tag{6.3}$$

and $\boldsymbol{D_{eff}} = \boldsymbol{J_m N^2 \eta} + \boldsymbol{D}$, $\boldsymbol{D_{eff}} \in \Re^n \times \Re^n$.

For compatibility with the formulation employed in Chapter 3, $\boldsymbol{u} = \boldsymbol{\tau_{total} N \eta}$, $\boldsymbol{u} \in \Re^n \times \Re^1$, while, for proper matrix dimensioning operation, the upper null submatrix in $\boldsymbol{B} \in \Re^n \times \Re^n$.

The optimal control of the well-known class of systems represented by Equation (6.2), such as the general robot manipulator under review, will be the subject of analysis hereafter.

6.2.1 A note about admissible controls

As discussed in Section 5.4.1, each joint of an industrial manipulator is normally driven by separate actuators and, therefore, it is only natural that any attempt to design a control strategy for the robot arm, and in particular an optimal controller, must be synthesised under the constraints imposed on the maximum torque bounds that can be independently exerted by the actuators [1]. Incidentally, the closed bounded admissible region of control inputs, $\mathcal{U}$, is widely considered fixed to minimise computational effort, as shown by Equation (3.13), and the overwhelming majority of published work in optimal control is based on these simplifying premises (see, for example, [3, 4, 5]).

While it is true that the absolute upper and lower bounds for each motor actuator under continuous operation are known design constants [2], it is also true that their instantaneous value for continuous operation is further limited by the maximum constant value of the driving actuator voltage $\pm V_m$, and the motor speed $\dot{q}_m$, as related by Equation (5.21) for a DC motor [6, 7]. The experimental test rig depicted in Figure 6.1 was set up to illustrate this limitation. Here, the upper-arm robot joint actuator was driven (open loop) by a constant maximum torque step input $\tau_{total}{}^+$, providing the results illustrated in Figure 6.2.

[1] An important characteristic too often overlooked by classical control analysis and design techniques.

[2] Traditionally referred to as *Continuous Stall Torque* or *Continuous Rated Torque* (Nm), in the motor technical specifications.

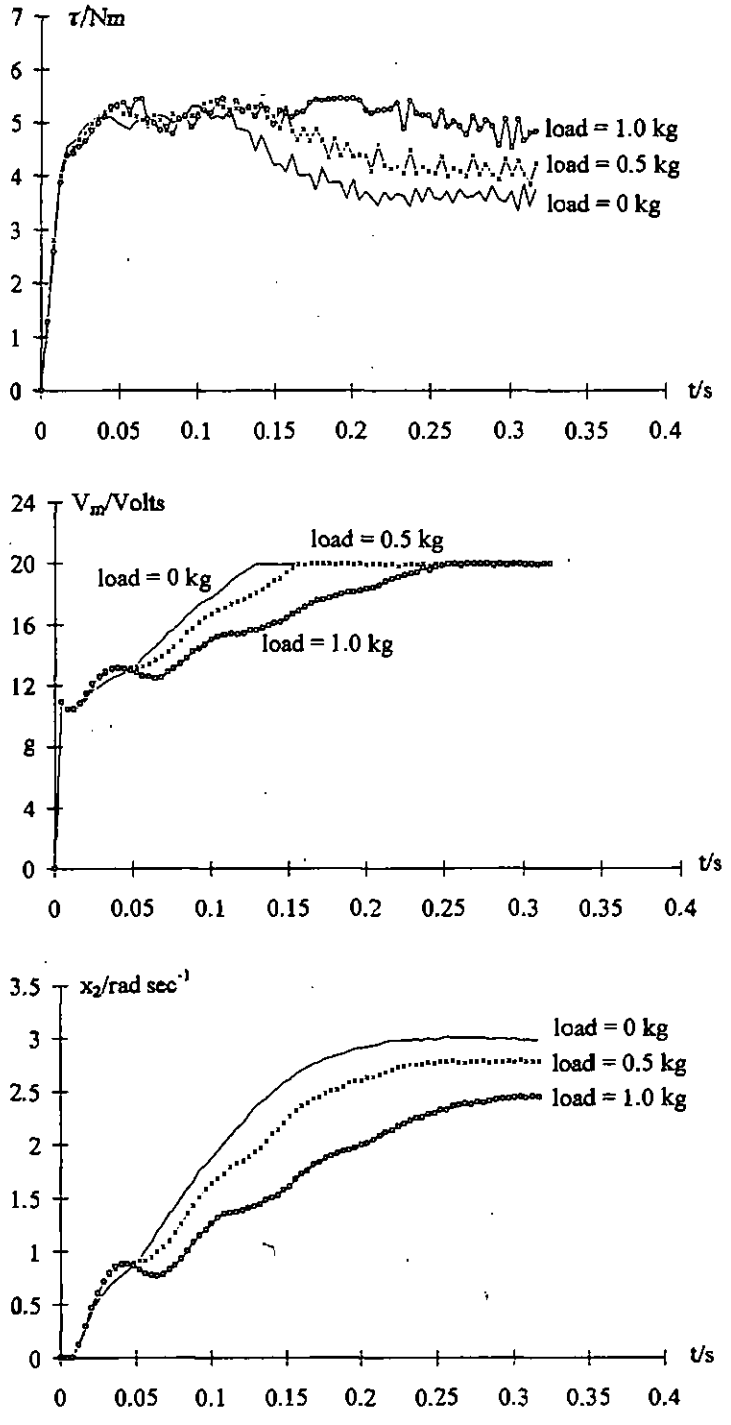

Figure 6.2: Upper-arm link (joint 3) torque (top), speed (bottom) and driving motor voltage (middle) response corresponding to maximum constant torque step.

The response link torque τ shown in the top graph is the result of applying Equation (5.21) to the commanded maximum torque, $\tau_{total}{}^{+}$, given the measured state (only link speed $\dot{q}_m$ in this case) which is depicted in the hottom graph. Note that the corresponding link speed x_2 is displayed in the graph instead of the motor speed, for uniformity with the link torque graph. If the desired torque exceeds the voltage/current capabilities of the driver, the limiting factor of the maximum driving voltage will set the new maximum torque τ_{total} by which the robot joint can be driven. This limiting voltage is depicted in the middle graph.

It can be seen how, for the maximum load specified by the manufacturer (1.0 kg) the torque exerted hy the actuator is also close to the continuous rated torque characteristic of the manipulator, correspondingly drawing the maximum current that the driver can deliver. This is achieved at the expense of slower maximum speed in comparison with carrying a lighter load, represented by the curves with load = 0.5 kg and no load at all. In those cases, the joint is accelerated faster to its maximum speed, which is also higher, but when reached, the exerted torque then drops to lower continuous values. The reason for this is that on reaching the turning point, i.e., when maximum driving voltage is reached (around 0.12 s for the unloaded case and 0.15 s for the 0.54 kg load case), no more current can be drawn (or torque exerted) for the given motor speed. It is precisely at this point that the control bounds should be adjusted to reflect the truly maximum torque available to the controller, which will be the specified maximum rated only when fully loaded, a situation not common during normal operation of a robot manipulator. It should be noted that joint speeds can be further increased on reaching this point as long as the driving voltage is within the constant physical limitations set by Equation (5.21).

In view of these facts, it is then clear that the set of admissible controls $\mathcal{U}$ for each joint actuator is then limited by the current speed and the constant maximum driving voltage, which can be mathematically formulated by the following state-dependant inequality constraints:

$$-U_i(x_{2_i}, -V_{m_i}) \leq u_i \leq +U_i(x_{2_i}, +V_{m_i}) \tag{6.4}$$

which show that, for each joint i, the true control constraint at every instant can be obtained hy satisfying Equation (5.21) on both sides of the inequality, given maximum possible driving voltage and current speed. In essence, then, Equation (6.4) is shifting the limiting control variable back to the physical driving voltage, another consequence of *reverse engineering* the actuator to design the controller in its natural *torque domain*, as discussed in Section 5.4.1. As a result of this, the controller design is simplified, while the real-time implementation is slightly more elaborated hy having to recalculate time-variable torque limits. Other authors [8, 9] have preferred to carry out the design of optimal controllers in the *electrical domain*, hence complicating the synthesis of the controller, but enhancing the clarity of the implementation because in that case the truly constant maximum driving voltage is the limiting factor directly employed in the design.

6.3 Problem Statement

The problem under review, initially outlined in Section 1.2.1, can be now summarized as follows:

Statement 6.1 *Utilising the state-space closed form of manipulator dynamics derived in Section 6.2, it is desired to find an allowable control $\boldsymbol{u}(t) \in \mathcal{U}$ that transfers the controlled robot plant to the desired region of the state-space and for which the IP $J(t)$, of the form described by (3.12), is minimised.*

A number of classical problems have been treated in the literature and can be formulated in terms of this fundamental control problem, depending on which parameters are weighted in the cost function $J(t)$. A compilation of some of these problems is listed in Table 6.1, to which the array of optimal control techniques introduced in Section 3.8 are applicable.

In the work described in this thesis, the time required by the manipulator to achieve the desired location, i.e., the optimal time problem, is considered. Mathematically, the problem of transferring the system from a given initial state at time t_i to a specified final state in minimum time can be expressed by ohtaining the minimum possible value of the following performance index:

$$J(t) = \int_{t_i}^{t_f} 1 dt \tag{6.5}$$

where t_f is the unspecified optimal final time.

Problem	Description
Terminal Control	Bring system as close as possible to given terminal state within a given period of time [10]
Minimum-time Control	Reach terminal state in the shortest possible period of time [11, 12]
The Regulator Problem	Keep system in equilibrium state so that IP is minimised [10]
The Tracking Problem [3]	Cause the state of the system to be as close as possible to a desired state time history. This is a generalisation of the regulator problem [10]
Minimum-energy Control	Transfer system from initial to final state with a minimum expenditure of control energy (fuel) [13]
Minimum-time-energy Control	Transfer system from initial to final state with a combined minimum expenditure of control energy in shortest allowable time [14, 15]
Minimum Acceleration Problem	Transfer system from initial to final state with a minimum expenditure in accelerating the system [16]

Table 6.I: Classical optimal control problems.

In view of these remarks, the minimum-time control problem becomes a special case of the more general optimal control problem defined above, which can be formulated as:

Statement 6.2 *Assuming the manipulator characterised by the state-space equations of motion (6.2), it is desired to find the optimal trajectory (and associated optimal control policy $\boldsymbol{u}(t) \in \mathcal{U}$) that the robot tool centre point (TCP), and consequently each joint, should follow to move (in joint space) from the initial position $\boldsymbol{x}_1(t_i)$ to the final position $\boldsymbol{x}_1(t_f)$, while minimising the performance index $J(t)$ given by (6.5).*

Despite the simplistic formulation of the problem (in essence, the equivalent to the system specifications of classical control analysis and design techniques), it is important to understand the complexity that the direct application of the MP brings to its solution. By applying Pontryagin's MP, Equation (6.2) should be substituted into the Hamiltonian functional, Equation (3.15), to derive the necessary conditions for optimality as given by the Hamiltonian canonical form, Equations (3.19) and (3.20). Having the problem prescribed by the two end-points, the necessary $2n$ boundary conditions required for the complete solution of the $2n$ first order differential equations describing the system, represented in matrix form by Equation (6.2), are readily available. However, as stated in Section 3.8.1, the analytical solution to this TPBV problem is extremely difficult in the majority of cases due to the inherent non-linearity and coupling in the manipulator dynamics.

It is beyond the scope of this dissertation to attempt any such numerical solution, and the reader is referred back to Section 4.3.3 where some of the off-line computational methods proposed in the literature are reviewed. The implicit problematic will be otherwise illustrated by finding an analytical solution to the relatively simple problem of the "Double Integrator" plant. The decision to study this particular example is not arbitrary. Beyond providing the means to illustrate the difficulty in designing optimal control strategies, it will be shown in Section 6.5 how the manipulator control problem being treated here can be reduced to the solution of a quasi-double integrator problem, hence providing a suitable approach to the on-line time-optimal control of a robot manipulator.

6.4 The Double Integrator Plant

The double integrator problem [10, 17] is a classic example to illustrate the use of the MP (and, in general, any other optimal control technique) in a variety of optimal control problems, such as the time-optimal control problem being considered here. The problem can be applied when the dynamic behaviour of the plant can be approximated by a second order single unitary-inertia system, and can

[3] Also referred to as the Servomechanism Problem.

therefore be described by the following set of ordinary differential equations:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= u \end{aligned} \tag{6.6}$$

It can be shown by applying Pontryagin's MP, that a necessary condition to transfer the system from a specified initial point $x(t_i)$ to a specified end-point $x(t_f)$ in minimum time is to let the control variable u take one or other of its extreme values, $+U \leq u \leq -U$, or simply $|u| \leq U$, which are considered fixed here in order to increase the clarity of the exposition. This requirement upon u can be easily demonstrated by deriving the Hamiltonian $\mathcal{H}$ defined by Equation (3.15), which for the double integrator plant can be described in the following form:

$$\mathcal{H} = -1 + p_1 x_2 + p_2 u \tag{6.7}$$

By inspecting Equation (6.7), it follows that for a given set of values p_1, p_2 and x_2, the Hamiltoninan $\mathcal{H}$ takes on its maximum value when the sign of the control signal u takes one or the other of its extreme values, $+U$ if p_2 is positive, $-U$ if p_2 is negative. Mathematically, this can be expressed by assigning a value to the optimal control signal u that has the same sign as that of p_2, and its magnitude is the maximum allowable value U:

$$u = U sgn(p_2) \tag{6.8}$$

where the *sgn* function is defined as

$$sgn(p_2) = \begin{cases} +1 & \text{if} \quad p_2 > 0 \\ 0 & \text{if} \quad p_2 = 0 \\ -1 & \text{if} \quad p_2 < 0 \end{cases} \tag{6.9}$$

This type of control is often referred to as *bang-bang* or *relay* control in the literature [10, 18]. It is apparent from Equation (6.8) that the solution to p_2 must be found first in order to implement this controller, and this is precisely where the major difficulty lies. By utilising the Hamiltonian canonical Equation (3.20), the costate equations can be derived, which for the simple case of the double integrator problem result in the following expressions:

$$\begin{aligned} \dot{p}_1 &= -\partial\mathcal{H}/\partial x_1 = 0 \\ \dot{p}_2 &= -\partial\mathcal{H}/\partial x_2 = -p_1 \end{aligned} \tag{6.10}$$

Thus the following set of 4 equations, 2 for the states and 2 for the adjoints must be solved simultaneously for the optimal control input :

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= U sgn(p_2) \\ \dot{p}_1 &= 0 \\ \dot{p}_2 &= -p_1 \end{aligned} \tag{6.11}$$

where u has been eliminated by substituting the optimal control action obtained from Equation (6.8). The four boundary conditions required for the complete solution of this optimisation problem are provided by the two specified initial and final values of the state variable $\boldsymbol{x}$.

As an alternative procedure to the customary numerical solution to the TPBV problem, it is relatively easy in this case to find an analytical solution. To this end, the form of the costate variables can be found by integrating (6.10), which result in:

$$\begin{aligned} p_1 &= a \\ p_2 &= -at + b \end{aligned} \tag{6.12}$$

where a and b are constants of integration which, in general, must be chosen to satisfy the boundary conditions on $\boldsymbol{x}$. However, because the costates are, in this case, independent of the state variables, the exact costate solution is unknown at this stage. It is interesting to note that if the initial conditions of the costate functions were known then the problem could be completely solved. Indeed, it is precisely a "good guess" of the costate's initial values the preferred approach that is employed by a large number of the numerical methods proposed in the literature for the solution of the problem (see Section 4.3.3 for more on this matter). Despite this shortcoming, a close examination of Equation (6.12) indicates

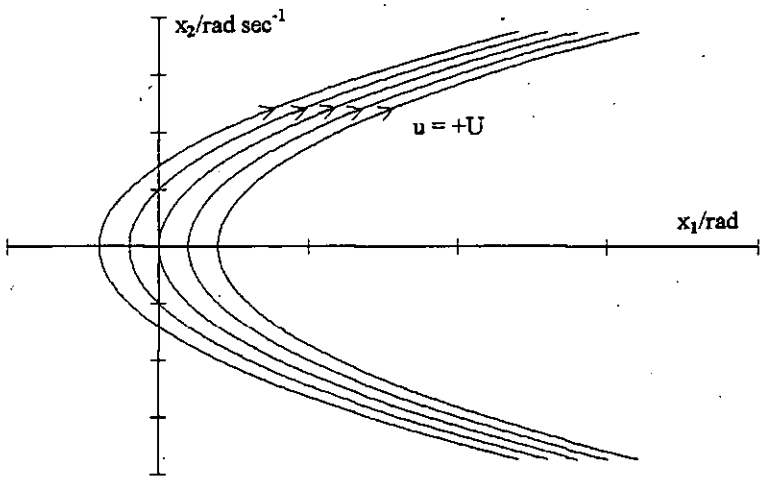

Figure 6.3: Double integrator state trajectories for $u = +U$.

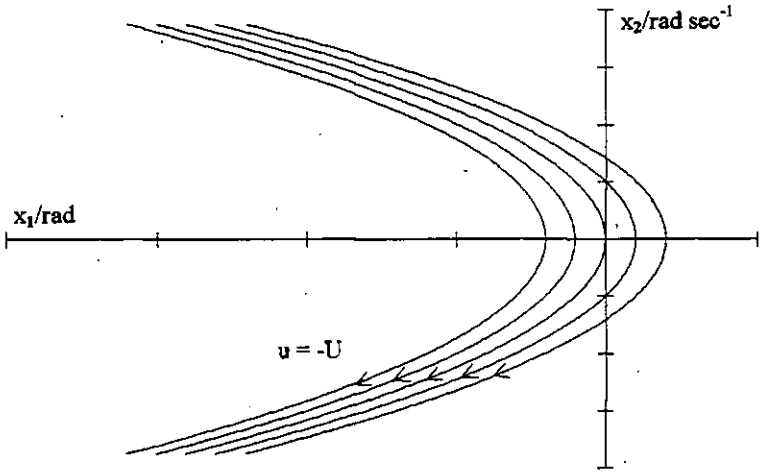

Figure 6.4: Double integrator state trajectories for $u = -U$.

that, as t varies over any range whatever, p_2 changes sign not more than once, depending on the value of the constants a and b. In view of Equation (6.8), therefore, it can be immediately seen that the optimal control input u, during the minimum-time transition from any specified initial state to any specified final state, takes on only the values $+U$ and $-U$, and changes sign at most once during the transition.

This conclusion can be ,then incorporated to study the *form* of the optimal trajectories. It will be first assumed that the desired final state $\boldsymbol{x}(t_f)$ is located at the origin $(0, 0)$. This is a straightforward transformation which can be easily achieved hy a trivial change of state variable values. Thus, segments of optimal trajectories can be found by integrating the state Equation (6.6) assuming one or the other of the the optimal control actions $\pm U$. The state trajectories of the system under the influence of $+U$ become:

$$x_2 = Ut + c \tag{6.13}$$

$$x_1 = \frac{U}{2}t^2 + ct + d \tag{6.14}$$

where c and d represent new constants of integration. Time t can be eliminated by squaring the first equation, and comparing with the result of multiplying Equation (6.14) by $2U$, to obtain:

$$x_2^2 = 2Ux_1 + \mathrm{e} \tag{6.15}$$

where $e = c^2 - 2Ud$, thus another constant. This equation can be rearranged to obtain the following equation for $u = +U$, the so-called "phase-plane" relationship between x_1 and x_2:

$$x_1 = \frac{1}{2U}x_2^2 - \frac{\mathrm{e}}{2U} \tag{6.16}$$

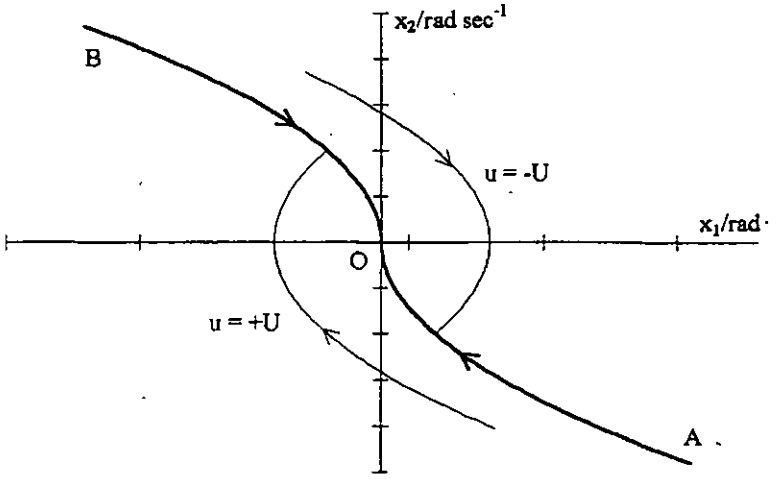

Figure 6.5: Double integrator state switching boundary and typical state trajectories.

For different values of e, Equation (6.16) defines the family of parabolas shown in Figure 6.3, where the arrows indicate the direction of increasing time. By similar reasoning, the relationship between x_1 and x_2 under the influence of u $= -U$ is of the form:

$$x_2 = -Ut + \check{c} \tag{6.17}$$

$$x_1 = \frac{1}{-2U}x_2^2 + \frac{\check{e}}{2U} \tag{6.18}$$

where $\check{e} = \check{c}^2 + 2U\check{d}$, and $\check{c}$ and $\check{d}$ are new constants of integration. The family of parabolas for Equations (6.17) and (6.18) under different values of $\check{e}$ can be seen in Figure 6.4.

Recalling that the optimal input takes on only the values $+U$ and $-U$, and changes sign at most once, several factors are apparent from Figures 6.3 and 6.4. Only one member of each solution family passes through any given point in state-space, and all optimum trajectories to $\boldsymbol{x}(t_f) = \mathbf{0}$ must eventually follow one of the two trajectories passing through the origin. Combining the two sets of trajectories into one, the family of time-optimal trajectories can easily be seen to be as depicted in Figure 6.5, where it is apparent that all optimum trajectories approach the origin from either the second or fourth quadrant. The two trajectories through the origin can be interpreted as a switching boundary in the phase-plane, as shown by the bold curve AOB in Figure 6.5. All initial states below the boundary curve require u $= +U$ until the switch curve is reached, followed by u $= -U$, which effectively slides the state along the switching curve towards the state origin. Initial states above the boundary require $u = -U$ followed by $u = +U$. An example of each of these conditions is also shown in Figure 6.5. It is interesting to note the symmetry of the optimal curve AOB with respect to both axes given the time-invariability of the double integrator plant. Therefore, it is easy to see that typical optimal trajectories consist of two consecutive segments, and only for initial conditions on the boundary AOB, no switching is required. In any case, it is possible to reach the origin from any initial state whatever with, at most, one sign change in the control input.

By assigning e $= \dot{e} = 0$ in Equations (6.16) and (6.18) respectively, the mathematical description of the switching curve AOB, as a function of the instantaneous state of the system, can be found to be:

$$x_1 = -\frac{1}{2U}x_2|x_2| \tag{6.19}$$

Hence, the time-optimal control law at any time t can be easily deduced in accordance with the following logical rules:

$$\mathrm{u}(t) = \begin{cases} +U(t) & \text{if } x(t) \text{ lies below } AOB \text{ or on } AO \\ -U(t) & \text{if } x(t) \text{ lies above } AOB \text{ or on } BO \end{cases} \tag{6.20}$$

From the above exposition, it should be conceptually clear now that even for a simple representation of the plant, such as that of the double integrator problem described by Equation (6.6), obtaining the optimal control law is not a straightforward procedure. The simple solutions for the state and costate

equations in this case allowed some general characteristics of the system to be drawn, which ultimately lead to the optimal control policy sought by an analytical method.

Unfortunately, although it is satisfying to think of minimum-time problems in this fashion, it is generally not a feasible approach to determine the solution of more dynamically complicated plants. It is increasingly difficult in those cases to extract amenable conclusions about the form of the state, costate and control patterns given their high non-linearity and coupling characteristics, as in the case of the manipulator plant considered here. In general [10]:

- For higher-order systems ($n \geq 3$), it is generally difficult, if not impossible, to obtain an analytical expression for the switching hypersurface.
- The procedure is generally not applicable to non-linear systems, because of the difficulty of analytically integrating the differential equations.

However, a near-optimal solution along the lines of that derived for the double integrator plant could also be adopted for a robotic arm, provided the manipulator dynamics could be approximated by a simpler and decoupled (in control input) system similar to (6.6) for each joint, while still being representative of the non-linear coupled plant dynamics.

Given the manipulator dynamics as represented by Equation (6.2), the intuitive methodology to achieve such a goal is that of linearising the manipulator dynamics around an operating point, usually the goal point, and then employ, if necessary, a linear transformation to a canonical form in which the controls are uncoupled, such as the Luenberger's transformation employed in [19]. When non-linearities are not severe, local linearisation in the neighbourhood of an arbitrary operating point is a valid strategy. Unfortunately, the manipulator control problem is, in general, not well suited to this approach because robot arms constantly move among widely separated regions of their workspace, such that no linearisation for all regions can be found [20].

The alternative solution is then to move the operating point with the manipulator as it moves, effectively resulting in a linear but time-varying system. Although the technique, referred to as *moving linearisation* by Craig [20], introduces a fair amount of additional computation caused by the substitutions and series expansion that need carrying out, it has the advantage of being far more representative of the time-variant non-linear plant dynamics, as they are linearised at each sample interval. However, an additional transformation, as mentioned above, is still required if a decoupled system is desired. Incidentally, the transformation, as proposed by some researchers in the past such as Kahn and Roth [19], neglects the time-variable inertia components of the linearised equation of motion, effectively reducing the overall robot arm system to a single time-invariant uncoupled double integrator for each joint.

In the following Section, an alternative strategy, to some extent similar to the latter solution, is presented to ultimately reach the same goal. That is, obtaining a plant structure for which a simple optimal control strategy, such as that of the double integrator problem, can be synthesised analytically. This can be made possible by the method of the *averaged dynamics*, first introduced by Kim and Shin [14], a strategy which can effectively approximate the coupled and non-linear manipulator dynamics by a piecewise linear simpler plant as it moves towards the goal point, but is still able to represent the overall dynamic characteristics of the plant. Consequently, the result is a *near-optimal* robot arm controller, yet applicable to real-time control environments. A number of issues are raised that justify the validity of the approach for coupled non-linear plants, in particular with regards to the suitability of the method for an on-line implementation in feedback form, as the practical results shown in the following Chapter will readily illustrate.

6.5 Near-Optimal Trajectory Planner/Controller Design

In order to understand the proposed strategy, it is useful to rewrite the manipulator equations of motion, as given by Equation (6.2), for each individual joint axis x_i. Assuming the same state variables, but now for each joint, i.e.,

$$\boldsymbol{x_i} = \begin{bmatrix} x_{i1} \\ x_{i2} \end{bmatrix} = \begin{bmatrix} q_i \\ \dot{q}_i \end{bmatrix} \qquad \text{for } i = 1 \ldots \text{n} \tag{6.21}$$

the 2-dimensional state-space system representation for each axis can be now extracted from Equation (6.2) as follows:

$$\dot{\boldsymbol{x}}_i = \boldsymbol{A}(\boldsymbol{x}) + \boldsymbol{B}(\boldsymbol{x})\boldsymbol{u} \qquad \text{for } i = 1 \ldots \text{n} \tag{6.22}$$

where

$$\boldsymbol{A}(\boldsymbol{x}) = \begin{bmatrix} x_{i2} \\ -\boldsymbol{D}_i^{-1}(\boldsymbol{x}_1)[\boldsymbol{H}(\boldsymbol{x}_1, \boldsymbol{x}_2) + \boldsymbol{G}(\boldsymbol{x}_1) + \boldsymbol{f}(x_{i2})] \end{bmatrix} \quad \boldsymbol{B}(\boldsymbol{x}) = \begin{bmatrix} 0 \\ \boldsymbol{D}_i^{-1}(\boldsymbol{x}_1) \end{bmatrix} \qquad \text{for } i = 1 \ldots n \tag{6.23}$$

and $\boldsymbol{D}_i^{-1} \in \Re^1 \times \Re^n$ represents the ith row of the inverse of the effective inertia matrix, $\boldsymbol{D}_{\boldsymbol{eff}} \in \Re^n \times \Re^n$. For proper matrix operation, the upper null component of $\boldsymbol{B}$ is now dimensioned as a vector in $\in \Re^1 \times \Re^n$.

It can be clearly seen from the structure of Equation (6.22) that motion at joint i is being affected by the control action on the other joints, a common characteristic of MIMO systems which too often impose limitations in the design of a suitable controller, as addressed by the Computed Torque control strategy described in Section 3.5.2. As well as couplings in the torque, via the cross inertia terms in $\boldsymbol{D}$, it is also easy to notice the non-linear interaxial reaction forces induced by velocity terms, $\boldsymbol{H}$, and gravity forces, $\boldsymbol{G}$, which also make a contribution to the dynamics of each joint.

It is interesting to note here that the application of non-linear cancellation control could, in theory, achieve the desired decoupling and linearisation of the plant. However, as shown in Figure 3.6, the trajectory to be tracked should be generated first, while, by the definition of the optimal *unconstrained* point-to-point motion problem presented here, there is no trajectory to be followed between the end-points.

In order to obtain a decoupled closed-form of the joint dynamics, Equation (6.22) can be rearranged into a form where the relationship between the acceleration of joint i and the actual actuator torque driving the joint becomes clear:

$$\dot{\boldsymbol{x}}_i = \begin{bmatrix} \dot{x}_{i1} \\ \dot{x}_{i2} \end{bmatrix} = \begin{bmatrix} x_{i2} & 0 \\ 0 & \alpha_i \end{bmatrix} \begin{bmatrix} 1 \\ u_i \end{bmatrix} + \begin{bmatrix} 0 \\ \beta_i \end{bmatrix} \qquad \text{for } i = 1 \ldots n \tag{6.24}$$

where

$$\alpha_i(\boldsymbol{x}_1) = D_{ii}^{-1}(\boldsymbol{x}_1) \quad \text{and} \quad \beta_i(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{u}) = \sum_{j=1, j\neq i}^{n} D_{ij}^{-1}(\boldsymbol{x}_1) u_j - \sum_{j=1}^{n} D_{ij}^{-1}(\boldsymbol{x}_1)[H_j(\boldsymbol{x}_1, \boldsymbol{x}_2) + G_j(\boldsymbol{x}_1) + f_j(\boldsymbol{x}_1)] \tag{6.25}$$

and D_{ij}^{-1} denotes the (i,j)th element of $\boldsymbol{D}_i^{-1}$.

Coefficients $\alpha_i(\boldsymbol{x}_1)$ and $\beta_i(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{u})$ are time-variant [4] non-linear functions of the manipulator position, velocity and control input, the latter collecting the coupling control effects from the other joints on joint i. However, Equation (6.24) could be regarded as an uncoupled time-invariant system for each robot arm axis by incorporating the following assumptions:

1. Providing the sampling interval, Δt, is small enough, the continuous control signal can be digitally approximated by a piecewise constant function which changes value only at time instants $t = 0, \Delta t, \ldots, (N-1)\Delta t$. This would effectively allow the variable inertial couplings in the joint at the current sample time t_c, included in $\beta_i(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{u})$, to be estimated using the last control input at time $t_c - \Delta t$, with a *minimal* overall error. This can be intuitively perceived from the characteristic of optimal control by which the value of the control is proved to be always on the boundary of the admissible region.

2. Additionally, non-linearities in $\alpha_i(\boldsymbol{x}_1)$ and $\beta_i(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{u})$ can also be incorporated into the model at *each* sample interval, provided information from the plant state is fed back to the controller. The consequence of this update, as far as the optimal controller is concerned, is that of essentially transforming the non-linear model of the manipulator joint, Equation (6.24), into a piecewise linear single-inertia plant for *each* feedback state, at *each* sample interval. Since the dynamic model is updated with state feedback information at the beginning of each interval Δt, the errors derived from the coupling and non-linear approximations can be, to a large extent, implicitly compensated.

[4] Not in the broad term of explicit time-dependency in the equation, but meaning "not constant" over time, given the state-dependency.

3. Furthermore, at present time, t_c, the optimal control strategy should be determined based on the dynamic behaviour of the manipulator over the period $[t_c, t_f]$. Yet the model dynamic coefficients are known for t_c (as a result of the update described above) and t_f, but not for the period in between. It is therefore necessary to find a way to describe the overall dynamic behaviour of the system for the remaining of the present motion on the basis of the current state and the final state. Some methods have been devised which propose an approximation of the dynamic coefficients. Particularly simple is the arithmetic average, as proposed in [14]:

$$[\overline{\alpha}_i(t_c), \overline{\beta}_i(t_c)] = \frac{[\alpha_i(t_c), \beta_i(t_c)] + [\alpha_i(t_f), \beta_i(t_f)]}{2} \qquad \text{for } i = 1 \ldots n \tag{6.26}$$

A more general form, similar to the one implemented in [13], has been adopted in which the overall dynamic behaviour of the manipulator is defined by a factor λ for each dynamic coefficient. This is shown in Equation (6.27). The inclusion of λ offers the flexibility to weight each boundary condition's dynamic performance separately in the estimated final value. The influence of this parameter will be studied in the following Chapter, Section 7.5.2.4, when controller simulation and implementation results are presented.

$$\begin{cases} \overline{\alpha}_i(t_c) = (1-\lambda)\alpha_i(t_c) + \lambda\alpha_i(t_f) & \lambda \in [0,1] \\ \overline{\beta}_i(t_c) = (1-\lambda)\beta_i(t_c) + \lambda\beta_i(t_f) & \text{for } i = 1 \ldots n \end{cases} \tag{6.27}$$

The combination of these conjectures can conveniently transform the original system into a suitable form for the analysis and design of the optimal controller, which was the original aim. Although a more detailed analysis of these approximations will be given in Section 6.5.2, it is important to understand the implications that the proposed transformations bring to the design of the controller:

1. Firstly, the decoupling approach is a simple methodology well suited to optimal control in a real-time environment which, despite the likelihood of introducing error, it is shown in Section 6.5.2 to be minor.

2. Secondly, the averaged dynamics adaption mechanism, while undoubtedly an approximation to the real plant, still preserves the overall *non-linear and time-varying* nature of the system, as given by the model of each manipulator joint, Equation (6.24). Hence, a completely different picture altogether from the traditional local linearisation method is obtained.

3. Thirdly, the simplicity of the averaged dynamics approach is shown to represent a good deal of computational savings in comparison with the moving linearisation and decoupling approach previously described.

In summary, thus, these assumptions allow the non-linear coupled manipulator plant to be regarded as a decoupled time-invariant system for each joint, updated in feedback form at the beginning of each sampling period with the manipulator present and final states (averaged dynamics) and decoupled by the last control action applied to the remaining joints. The resulting model equation for each link at each update interval can be now described by:

$$\dot{x}_i = \begin{bmatrix} \dot{x}_{i1} \\ \dot{x}_{i2} \end{bmatrix} = \begin{bmatrix} x_{i2} & 0 \\ 0 & \overline{\alpha}_i \end{bmatrix} \begin{bmatrix} 1 \\ u_i \end{bmatrix} + \begin{bmatrix} 0 \\ \overline{\beta}_i \end{bmatrix} \qquad \text{for } i = 1 \ldots n \tag{6.28}$$

Hence, a similar model to that represented by Equation (6.24) is obtained where the coefficients $\overline{\alpha}_i$ and $\overline{\beta}_i$ are now constant parameters representative of the overall manipulator dynamics, as derived from the averaged dynamic procedure given by Equation (6.27). Incidentally, the problem is then reduced to the solution of a quasi-double integrator problem for each joint, that must be solved in real-time at each sample interval. The structure of the optimal controller for such a system is described next.

6.5.1 Controller structure

The derivations described in Section 6.4 for the double integrator plant can now be extended to solve the trajectory planning and control of the uncoupled quasi-double integrator manipulator plant problem, as represented by Equation (6.28). The Hamiltonian for each joint can be now obtained as:

$$\mathcal{H} = -1 + p_1 x_{i2} + p_2 \overline{\alpha}_i u_i + p_2 \overline{\beta}_i \tag{6.29}$$

Since $\overline{\sigma}_i$ represents the joint effective inertia, which is always positive [20], a close examination of $\mathcal{H}$ shows that, as with the double integrator plant, for any given value of p_1, p_2, x_{i2}, $\overline{\alpha}_i$ and $\overline{\beta}_i$, the Hamiltonian takes on its maximum value when the magnitude of the control signal is at the boundary, and has the same sign as p_2, i.e.,

$$\mathrm{u} = U sgn(p_2) \tag{6.30}$$

where the *sgn* function is defined by Equation (6.9). Following the mechanism previously described for the double integrator, the same costate variables are derived, as given by Equation (6.12), and therefore the same conclusion is drawn: the control input changes sign at most once during the current motion, given the *current* dynamics of Equation (6.28). The expression for the phase parabolas can be found without difficulty by integrating Equation (6.24) after substituting the control action by its maximum value, and then eliminating time from the expression. For $\mathrm{u} = +U$:

$$x_{i2} = (\overline{\alpha}_i U + \overline{\beta}_i)t + c \tag{6.31}$$

$$x_{i1} = \frac{1}{2(\overline{\alpha}_i U + \overline{\beta}_i)} x_{i2}^2 - \frac{e}{2(\overline{\alpha}_i U + \overline{\beta}_i)} \tag{6.32}$$

where c and e are constants. Equally, the phase-plane trajectory under the influence of $\mathrm{u} = -U$ can be obtained as:

$$x_{i2} = (-\overline{\alpha}_i U + \overline{\beta}_i)t + \check{c} \tag{6.33}$$

$$x_{i1} = \frac{1}{2(-\overline{\alpha}_i U + \overline{\beta}_i)} x_{i2}^2 - \frac{\check{e}}{2(-\overline{\alpha}_i U + \overline{\beta}_i)} \tag{6.34}$$

where $\check{c}$ and $\check{e}$ also denote constants.

When the double integrator problem was presented, it was shown how the orientation of the parabolas relative to x_1 was uniquely determined by the control action: upright along x_1 when $+U$ (Figure 6.3), and inverted when $-U$ (Figure 6.4). Moreover, given the symmetry of the parabolas in both cases, a unique formula for the switching curve sufficed (Equation (6.19)). For the quasi-double integrator problem, a different scenario arises. As can be seen from Equations (6.32) and (6.34), for a given state, the orientation of the parabolas is dependant not only on the value of the control variable, but on the combined value $\overline{\alpha}_i \mathrm{u}_i + \overline{\beta}_i$. Hence, for a given set of values of $\overline{\alpha}_i$ and $\overline{\beta}_i$, the resulting parabolas could, in principle, be oriented either way under the action of any of the two possible control actions. A little thought will show that, in real terms, that is not the case. For a given state of the system, the response for all the possible cases of $\overline{\alpha}_i$ and $\overline{\beta}_i$ can be analysed under the influence of applying both control actions, that is:

$$\text{case } +U \text{ is applied} = \begin{cases} \overline{\alpha}_i U \le \overline{\beta}_i \Rightarrow \overline{\beta}_i \ge 0 \Rightarrow \text{parabola upright} \\ \overline{\alpha}_i U > \overline{\beta}_i \Rightarrow \begin{cases} \overline{\beta}_i \ge 0 \Rightarrow \text{parabola upright} \\ \overline{\beta}_i < 0 \Rightarrow \text{parabola upright} \end{cases} \end{cases}$$

$$\text{case } -U \text{ is applied} = \begin{cases} -\overline{\alpha}_i U < \overline{\beta}_i \Rightarrow \begin{cases} \overline{\beta}_i \le 0 \Rightarrow \text{parabola inverted} \\ \overline{\beta}_i > 0 \Rightarrow \text{parabola unknown} \end{cases} \\ -\overline{\alpha}_i U \ge \overline{\beta}_i \Rightarrow \overline{\beta}_i \le 0 \Rightarrow \text{parabola inverted} \end{cases} \tag{6.35}$$

It can be seen that, except in one case, the result of applying the maximum and minimum control action provide the same intuitive result as that of the double integrator problem, i.e., $+U$ generates upright parabolas where speed increases over time as a result of the positive control action, and $-U$ inverted paraholas, thus reducing speed over time as a result of injecting a negative control action. The exceptional case arises because two alternatives appear, depending on whether $|-\overline{\alpha}_i U| \ge \overline{\beta}_i$ or $|-\overline{\alpha}_i U| < \overline{\beta}_i$. The former case generates inverted parabolas, just like the general case of appying $-U$. However, the latter implies that the contribution of all forces other than the effective inertial joint force can be larger than the purely inertial torque under the application of $-U$, in turn generating an upright parabola under its influence. Although this effect can physically happen, depending on the significance of the dynamic parameters, the contribution of the inertial parameter has been determined to be the most significant of all parameters [21], even more so under the effect of the maximum input control action. Therefore this case would rarely occur, if at all. But even in those cases when it could happen, the effect of applying $-U$ would, despite still increasing the speed of the manipulator joint, take it to obvious slower thresholds, which is not a desirable objective for optimal motions. Therefore, $+U$

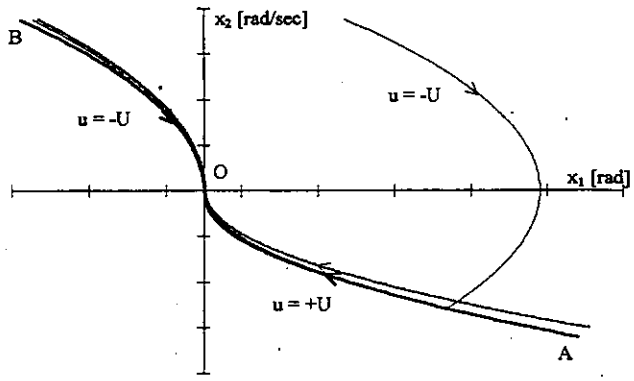

Figure 6.6: Quasi-Double Integrator State Switching Curve.

will be the preferred strategy on such occasions. On the basis of these facts it can be concluded that, as for the double integrator problem, the result of applying $+U$ generates phase parabolas as shown in (Figure 6.3), while those in (Figure 6.4) are the result of $-U$. However, for a given set of values of $\overline{\alpha}_i$ and $\overline{\beta}_i$, the parabolas resulting from applying the two possible controls need not he symmetric anymore. In fact, they will only be when $\overline{\alpha}_i U + \overline{\beta}_i = -\overline{\alpha}_i U + \overline{\beta}_i$, which is rarely the case.

As before, by assigning $e = e' = 0$ in Equations (6.32) and (6.34) respectively, the formulation for the switching curves that all optimal trajectories must eventually follow to pass through the origin can be easily found. However, now, the value of $\overline{\alpha}_i U + \overline{\beta}_i$ and $-\overline{\alpha}_i + \overline{\beta}_i$ need not necessarily be of equal magnitude and opposite sign, so that the optimal switching curve at each time t can not be described by a unique equation, and therefore the control law needs to be considered for two separate cases depending on the velocity of the joint:

$$\begin{cases} \text{if } x_{2i}(t) \geq 0 \\ \qquad u_i(t) = \begin{cases} U & \text{if } x_{1i}(t) < \dfrac{x_{2i}^2(t)}{2(-\overline{\alpha}_i U + \overline{\beta}_i)} \\ -U & \text{otherwise} \end{cases} \\ \text{if } x_{2i}(t) < 0 \\ \qquad u_i(t) = \begin{cases} U & \text{if } x_{1i}(t) \leq \dfrac{x_{2i}^2(t)}{2(\overline{\alpha}_i U + \overline{\beta}_i)} \\ -U & \text{otherwise} \end{cases} \end{cases} \qquad \text{for } i = 1 \ldots n \qquad (6.36)$$

The behaviour of such a control law is depicted by the near-optimal state switching curves shown in Figure 6.6. The phase-plane graph represents the evolution of the system state, along with some of the switching curves generated during a typical run, and clearly shows the two distinctive features of the new near-optimal control law :

- The parabolas that define the switching curves (AOB curves) are not time-invariant anymore, but they need to he recalculated at each sample time, according to Equation (6.36). While only a number of the curves generated are shown in Figure 6.6 for clarity, it can he seen that they are not distant from each other, hence proving that the averaged dynamics method can accomodate for the changing dynamics, on which the computation of the switching curves is grounded as given by Equation (6.36).
- The application of the two optimal control actions, U and $-U$, generates non-symmetric optimal curves for positive and negative speeds, as discussed above. This is opposed to the symmetric parabolas shown in Figure 6.5 for a linear plant.

The evolution of the system state for the case when the initial state is located in the first quadrant is also shown in Figure 6.6 with a thin line.

The resulting algorithm that yields the minimum-time trajectory and control action the robot should follow to move, unconstrained, from an initial to a final state can be formulated as follows:

1. Derive the manipulator dynamic model in the form of Equation (6.24).
2. Given desired initial and final states, compute the corresponding control actions from Equation (5.15). The initial control action will act as the estimated control input into the algorithm.

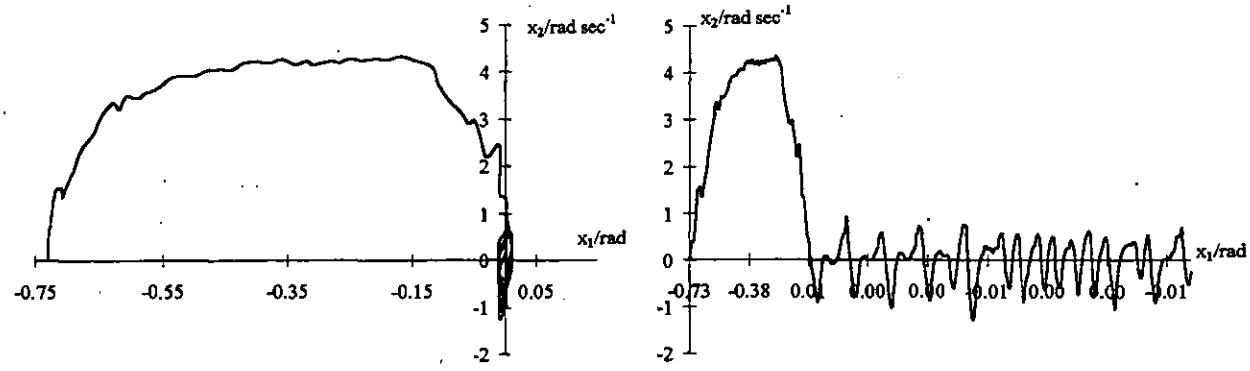

Figure 6.7: Chattering effect. Phase-plane (left) and expanded phase-plane representation.

3. Given current state, calculate the maximum permissible control actions which satisfy inequality (6.4).

4. Calculate current state's dynamic behaviour for each joint simultaneously. This is described by $\alpha_i(\boldsymbol{x_1})$ and $\beta_i(\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{u})$ in Equation (6.25), where the coupled inertial terms are decoupled by the effect of approximating the current control action exerted by the joint with the last control applied, as formerly described.

5. Average the dynamic coefficients according to Equation (6.27) to achieve the overall dynamic performance of the manipulator, somewhat representative of the whole motion from the current state to the final goal state. A constant value of 0.4 for the coefficient λ has been found to provide the best experimental results, as will be shown in Section 7.5.2.4, thus assigning slightly more importance to the desired current state in the overall dynamics.

6. Employ the current estimated dynamic model to update the switching curves according to the control law described by Equation (6.36) and compute the optimal control action.

7. Calculate the driving voltage corresponding to the demanded near-optimal input torque according to Equation (5.21).

8. Go back to point 3 if final state has not been reached.

The application of this algorithmic controller will drive the manipulator to the origin of the phase-plane as desired. However, a well-known problem with any bang-bang method is control chattering in the vicinity of the target state caused by frequent switchings of the control input, as shown in the phase-plane representation of Figure 6.7, left. The graph on the right is another representation of the same curve, where the positional state has been expanded with time to clearly show the chattering around the origin. Different alternatives to alleviate this undesirable effect have been proposed in the literature, e.g., the use of a smoothing function [8] or switching to a linear controller when the manipulator is within a prescribed range of the goal steady-state [14]. In the work presented here the possibility of applying a Feedforward Controller has been tested with successful results, as will be presented in the following Chapter. This means that, when the phase portrait reaches some specified state threshold near the origin, the corresponding control is calculated according to Equation 3.8 around the desired end position, effectively exerting the steady-state holding torque of the joint. Therefore, point 8 of the algorithm is further extended to compare current state with target state. If the difference is less than some state bounds specified by the user, set experimentally to 0.27 rad/s and 0.027 rad for best results, the control switches into a Feedforward Controller. Motion is completed when all the joints have reached the desired goal position.

6.5.2 Controller analysis

Having derived the algorithmic structure of the unconstrained near-optimal controller, a number of issues about its design should be taken into consideration.

6.5.2.1 Approximations-related issues

It has been consistently stated that the main reason behind the various approximations taken to solve the optimal control of coupled and non-linear plants, such as robot manipulators, lies in obtaining a *practical* solution to the problem. The implications on the optimality of the solution derived from the assumptions described in page 82 are twofold:

1. By employing the old $(t-\Delta t)$ control action in calculating the dynamics of the plant for the current (t) state, no error is introduced in the calculation of $\alpha_i(\boldsymbol{x_1})$, but the inertial coupling terms in $\beta_i(\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{u})$ are somewhat approximated to their real optimal values at time t. Incidentally, this may introduce an error in the calculation of the switching curves, as this takes place on the basis of the current dynamics of the plant. However, the application of bang-bang control, as in the case of time-optimal motion, implies that *only* when there is a switch in the control input as a result of the optimal switching curves, would an error in the coupling term arise. Under normal operation of a robotic arm, particularly for unconstrained point-to-point motions, there are *usually* two changes in control, that is, during the acceleration and deceleration phase (see, for instance, [18], [19], [22] or [23], and also results in the following Chapter), therefore, the influence of such an error can be deemed insignificant.

2. The other issue is that of estimating manipulator dynamics by the method of the averaged dynamics. Similarly to any linearisation method, errors are introduced which, for the case of the optimal control of a robot manipulator problem, are particularly difficult to measure given the complexity in obtaining an exact analytical [5] or numerical solution. However, unlike (local) linearisation techniques, the averaged dynamics method preserves the overall non-linearities of the plant by representing the dynamics as a piecewise linear time-invariant plant around the current state at the current iteration. This permits for the error generated by the averaging technique at the current sampling time t to be implicitly compensated at the following iteration $t + \Delta t$ by the new state feedback, essentially making the system less sensitive (more robust) to parameter variations, showing once again the viability of the controller for on-line purposes.

6.5.2.2 Feedback issues

Having referred to the robustness introduced by the state feedback feature of the near-time-optimal algorithm, it is important to understand the general advantage that closing the loop brings to any control strategy. Feedback operation makes use of the most recent information on the state of the plant, in this case to calculate the switching surfaces at each iteration, and therefore the optimal control. As a result, if a disturbance was to occur within the feedback system, the controller would optimally operate on the latest measurement, rather than attempting to return to the original trajectory or follow a pre-programmed input, as would he the case for open-loop systems. This is one of the main objections to optimum trajectories obtained open-loop by numerical methods, since once the system deviates from the optimal trajectory, subsequent motion no longer follows the precalculated optimal trajectory, nor adapts to that change. Moreover, the resulting numerical solution is valid only for the specific problem being considered, and it is very difficult to extend the result and obtain a general solution which can be used for other problems [2].

6.5.2.3 Co-ordinated motion issues

Optimal control problems along a specified path lend themselves naturally to co-ordinated motions, i.e., motions where all the joints start and stop at the same time. This is because the problem can then be parameterised by a unique scalar parameter measuring the arc-length of the path, which when solved, determines the torques for all the joints of the robot arm, as described in Section 4.3.1. When a numerical solution to the unconstrained optimal problem is attained, no decoupling is attempted since it is not an issue for the solution, and a synchronised motion is easily achieved. However, when the motion of the manipulator is described by each individual axis, as is the case here, final times may not he the same for all joints. In fact, they rarely are [14, 19]. Notice, however, that when the joint reaches its final position, it is held there by the steady-state torque calculated from the Feedforward Controller.

[5] Very often impossible for high order non-linear systems.

Hence, possible displacements from the desired end position due to coupling effects from other joints still at motion are considered in feedback form by the linear PD portion of the Feedforward Controller.

6.5.3 Stability analysis

Even though the general propositions of stability and performance analysis of linear systems have been extensively developed in the literature, giving way to well established techniques, only a limited number of tools have been made available for the analysis of non-linear systems, otherwise an active area of research [24]. Furthermore, given the fundamental properties of non-linear systems, no uniform approach to their analysis is possible, and available techniques depend on such factors as the severity of the non-linearity, the order of the system under consideration and/or the form of the input, hence rendering them applicable to the analysis of *specific* systems under *specific* conditions. Some of the techniques available include:

- The Describing Function approach [1], which is applicable to non-linear systems of any order, but assumes the input to be of sinusoidal form. Also, while the describing functions of single non-linearities, such as hysteresis or backlash, are relatively easy to obtain (in fact, they can be looked up in tables), if a system contains more than one non-linearity, they must be lumped together and an overall describing function obtained. This is by no means a straightforward task for complex non-linear systems such as robotic manipulators.

- Popov's Frequency Domain Method [25] provides a sufficient condition for asymptotic stability of time-invariant non-linear control systems but, as the describing function, the method is restricted to single-loop plants which can be decomposed in a linear process and a non-linear element (such as a saturation or a relay). In fact, the key element of the technique is based on characteristics derived from the linear portion of the system.

- The Generalised Circle Criterion [2] is essentially an extension of Popov's method to account for time-variable non-linearities also, but the same restrictions apply.

- A very powerful and elegant technique to determine the steady-state stability of non-linear systems based on generalisations of energy notions is Lyapunov Stability Analysis [2, 20]. It should perhaps also be noted that the original Russian text by Lyapunov (dating back to 1892) is now available in an English translation [26]. Lyapunov theory has become increasingly prevalent in robotic research publications [27, 28]. The reason is the ease of application compared to the approaches described above, mainly because new control algorithms for non-linear systems can be proved (asymptotically) stable by choosing the appropriate definite scalar function of the state variables with certain required properties. On the other hand, although some general functions are widely employed with successful results, choosing the right one is still a matter of experience.

Unfortunately, the structure of the control algorithm proposed in this work, (Equation (6.36)) poses a major drawback in the stability analysis with these techniques. The assumptions undertaken in its design are in part responsible for that, but it is mainly its variable structure which prevents the controller from being directly accommodated in the feedback loop for stability analysis purposes. Although further work could be dedicated to the tedious pursuit of the right Lyapunov function(s) to provide a general answer, the methodology employed in the design of the controller lends itself naturally to the graphical Phase-Plane technique [1, 2] of stability analysis of first and second order systems, particularly useful for systems with any number of non-linearities.

The technique generates a *phase portrait* of the system by studying the transient response of the non-linear control system to an external input under different initial conditions, as shown in Figure 6.5. This is accomplished either by specifically solving the non-linear equation of the plant, or by phase-portrait sketching methods such as that of the Isoclines [2] when no analytical solution can be obtained. It can be intuitively deduced from the general concept of stability that, if the phase trajectory followed by the system approaches the vicinity of the origin, the system can be regarded as stable. Given the design specifications followed during the synthesis of the near-optimal manipulator controller, it can be guaranteed that for any given state, the trajectory followed by the manipulator will always be that of approaching one of the two possible switching curves (depending on whether velocity is positive or negative at each particular instant) as described by Equation (6.36), and then sliding down to the state

origin, continuously applying the necessary (near-optimal) control action to drive the plant back to the switching trajectory under any possible disturbance. In view of these facts, it can then be concluded that the actual specifications under which the controller is designed provide the necessary conditions for the asymptotic stability of the plant. The author is nevertheless aware that a more rigorous mathematical proof would be a valuable asset, and that is one of the suggested areas for further work surveyed at the end of Chapter 8.

An additional contribution to the stability of the system is also the choice of sampling rate, explained in more detail in Section 7.4. This was set to a value of 4 ms (i.e., a frequency of 250 Hz), which provided a much greater rate than 20 times the 10 Hz mechanical time constant of the arm links, hence minimising any deterioration of the controller due to sampling [29].

A final check on the stability and performance analysis of the controller will be provided by simulation in the following Chapter. This is found particularly necessary in this case to demonstrate stability conclusively. Moreover, it will aid in overcoming such factors as possible uncertainties regarding the validity of the assumptions made in the previous Section, and the difficulties caused by the manipulator plant complexity which can not be studied by any other analytic method.

6.6 Summary and Discussion

A strategy has been presented in which the role of manipulator dynamics in trajectory planning and control is investigated within the context of optimal control. Given the non-linear and coupling characteristics of robot manipulators, some hypothesis needed to be undertaken if a numerical solution to the optimal control problem was to be avoided. The assumptions taken about the manipulator dynamics along the point-to-point motion have resulted in a regulating algorithm, suitable for real-time implementation, which maximises the capabilities of the device, hence improving the manipulator time response.

Although the assumptions taken result in the control of the manipulator in a near-optimal fashion, it has been confirmed in Section 6.5.2.1 that the end result will always lie very close to the true optimal solution. It is important to realise that the proposed controller does not force the manipulator to follow a prescribed trajectory. It moves the arm to a goal point along a collision-free trajectory virtually specified by the manipulator dynamics, and then regulates the position there with the aid of a Feedforward Controller. While the main justification of the hypothesis undertaken for its design was to aid in deriving an *on-line* analytical solution to the problem, on the other hand successfully implemented as shown in the succeeding Chapter, nothing prevents the derived trajectory to be employed as an off-line reference trajectory to be controlled by any other more traditional algorithm if so desired. While this betrays the main goal of the strategy presented, it should still provide better results than the schemes implemented on common industrial manipulators.

The stability of the algorithm has also been established by its own design methodology with the assistance of phase-plane techniques. However, as emphasised in Section 6.5.3, given the particular characteristics of the controller and the lack of an all embracing theory equivalent in its widespread use to the theory of linear systems, the ultimate check on stability will be given by the simulation of the plant in the subsequent Chapter.

References

[1] Atherton D.P. *Nonlinear Control Engineering.* Van Nostrand Reinhold Co. Ltd., (1975).

[2] Shinners S.M. *Modern Control System Theory and Design.* John Wiley & Sons, Inc., (1992).

[3] Pfeiffer F. and Johanni R. A concept for manipulator trajectory planning. *IEEE Journal of Robotics and Automation*, **RA-3**(2):115–123, April (1987).

[4] Geering H.P., Guzella L., Hepner S.A.R., and Onder C.H. Time-optimal motions of robots in assembly tasks. *IEEE Transactions on Automatic Control*, **AC-31**(6):512–518, June (1986).

[5] Shiller Z., Chang H., and Wong V. The practical implementation of time-optimal control for robotic manipulators. *Journal of Robotics & Computer-Integrated Manufacturing*, **12**(1):29–39, (1996).

[6] Verl A. and Bodson M. Torque maximization for permanent magnet synchronous motors. In *Proceedings of the European Control Conference*, July (1997). Session TH-A G5, paper number 525 (in CD-ROM).

[7] Shin K.G. and McKay N.D. Selection of near-minimum time geometric paths for robotic manipulators. *IEEE Transactions on Automatic Control*, **AC-31**(6):501–511, June (1985).

[8] Sattar T.P., Krejnin G.V., Smelov L.A., and Bolotnik N.N. Time-optimal control strategy for robotic-manipulator. theoretical and practical research. In *Proceedings of the 11th ISPE/IEE/IFAC International Conference on CAD/CAM, Robotics & Factories of the Future*, pages 872–877, Pereira, Colombia, August (1995).

[9] Tarkiainen M. and Shiller Z. Time optimal motions of manipulators with actuator dynamics. In *Proceedings of the IEEE Conference on Robotics and Automotion*, volume 2, pages 725–730, (1993).

[10] Kirk D.E. *Optimal Control Theory: an introduction.* Prentice-Hall Inc., (1970).

[11] Cao B. and Dodds G.I. Time-optimal and smooth joint path generation for robot manipulators. In *Proceedings of Control'94*, pages 1122–1127, March (1994).

[12] Sundar S. and Shiller Z. Constrained optimization of multi-degree-of-freedom mechanisms for near-time- optimal motions. *Journal of Mechanical Design. Transactions of the ASME*, **116**:412–418, June (1994).

[13] Wiens G.J. and Berggren M.J. Suboptimal path planning of robots: minimal nonlinear forces and energy. *Journal of Dynamic Systems, Measurement, and Control. Transactions of the ASME*, **113**:748–752, December (1991).

[14] Kim B.K. and Shin K.G. Suboptimal control of industrial manipulators with a weighted minimum time-fuel criterion. *IEEE Transactions on Automatic Control*, **AC-30**(1):1–10, January (1985).

[15] Wapenhans H., Hölzl J., Steinle J, and Pfeiffer F. Optimal trajectory planning with application to industrial robots. *The International Journal of Advanced Manufacturing Technology*, 9:49–55, (1994).

[16] Kano H. and Takayama K. Smooth trajectory control of robotic manipulators based on minimum acceleration criterion. *Advanced Robotics*, 5(2):147–164, (1991).

[17] McCausland I. *Introduction to Optimal Control.* John Wiley & Sons Inc., (1969).

[18] Weinreb A. and Bryson Jr.A.E. Optimal control of systems with hard control bounds. *IEEE Transactions on Automatic Control*, **AC-30**(11):1135–1138, November (1985).

[19] Kahn M.E. and Roth B. The near-minimum-time control of open-loop articulated kinematic chains. *Journal of Dynamic Systems, Measurement, and Control. Transactions of the ASME*, 93(3):164–172, September (1971).

[20] Craig J.J. *Introduction to Robotics: mechanics and control.* Addison-Wesley Publishing Company Inc., second edition, (1989).

[21] Corke P.I. An automated symbolic and numeric procedure for manipulator rigid-body dynamic significance analysis and simplification. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1018–1023, April (1996).

[22] Chen Y.C. Solving robot trajectory planning problems with uniform cubic b-splines. *Optimal Control Applications & Methods*, 2:247–262, (1991).

[23] Singh S. and Leu M.C. Optimal trajectory generation for robotic manipulators using dynamic programming. *Journal of Dynamic Systems, Measurement, and Control. Transactions of the ASME*, **109**:88–96, June (1987).

[24] Isidori A. *Nonlinear Control Systems.* Springer-Verlag, third edition, (1995).

[25] Popov V.M. Absolute stability of nonlinear systems of automatic control. *Automatic Remote Control (USSR)*, **22**:857–875, (1961).

[26] Lyapunov A.M. *On the general probem of stability of motion.* Taylor and Francis, (1992). Translated and edited by A.T. Fuller.

[27] Arimoto S. and Miyazaki F. Stability and robustness of pid feedback control for robot manipulators of sensory capability. In *Proceedings of the First International Symposium on Robotics Research*, pages 783–801. MIT Press, August (1984).

[28] Slotine J.J. and Li W. On the adaptive control of robot manipulators. *International Journal of Robotics Research*, **6**(3):49–59, (1987).

[29] Fu K.S., Gonzalez R.C., and Lee C.S.G. *Robotics: control, sensing, vision, and intelligence.* McGraw-Hill Book Co., (1987).

Chapter 7

Implementation and Results

7.1 Introduction

The effectiveness of the control strategy detailed in the previous Chapter is demonstrated here via the simulation and implementation of the algorithm. Theoretical analysis and computer simulations of the near-optimal controller are important but not sufficient. The ultimate justification of the value and applicability of the new controller lies in its actual hardware implementation.

In pursuit of this goal, a test rig was designed and constructed around the CRS A251 industrial manipulator. Much of the existing robot controller architecture, i.e., the driving hardware and feedback signalling, were reused. This scenario is presented in Section 7.2. While the electro-mechanical characteristics of the CRS A251 Small Industrial Robot System were extensively reviewed and studied in Chapter 5, the most important features of the existing CRS A251 Robot System Controller (RSC) are examined in Section 7.2.1. Off-the-shelf data acquisition and newly designed communication interface boards, required to enable control of the plant using an external personal computer (PC), are described in Section 7.2.2.

A brief description of the software (SW) and hardware (HW) simulation environment is first described in Section 7.3. Given the similarities between the simulation and experimental controller SW coding (both developed in structured ANSI C), a more detailed study of the actual controller program has been restricted to the latter, thus avoiding unnecessary redundancies. This is presented in Section 7.4. Additionally, some of the real-time problems encountered and the solutions adopted are exposed, and a data flow schematic diagram is also provided to help understand the coding of the controller (provided in Appendix B).

The set of experimental and simulation results are compiled in Section 7.5. This is initiated by a definition, in Section 7.5.1, of the criteria to be followed in analysing and evaluating the output data. Given the high volume of data generated from simulations, real-time experiments and measurements from the existing PID controller for comparison purposes, only two of the eight cases analysed are presented in full in Sections 7.5.2.1 and 7.5.2.2. The results from the remaining cases, reduced to tabular form, are discussed in Section 7.5.2.3. Furthermore, an in-depth analysis of the influence of the dynamic parameter λ in the overall structure of the algorithm is also undertaken in Section 7.5.2.4. This is then followed by a brief discussion of the results in Section 7.6, which finally concludes the Chapter.

7.2 Experimental Setup

A schematic description of the experimental environment is shown in Figures 7.1 and 7.2. The bottom diagram, Figure 7.2, illustrates the development of new measurement and control equipment, which was added to the standard industrial controller shown in the top diagram - Figure 7.1 - to evaluate the new motion strategies. Following this scheme, the existing and the new HW/SW configurations are presented next.

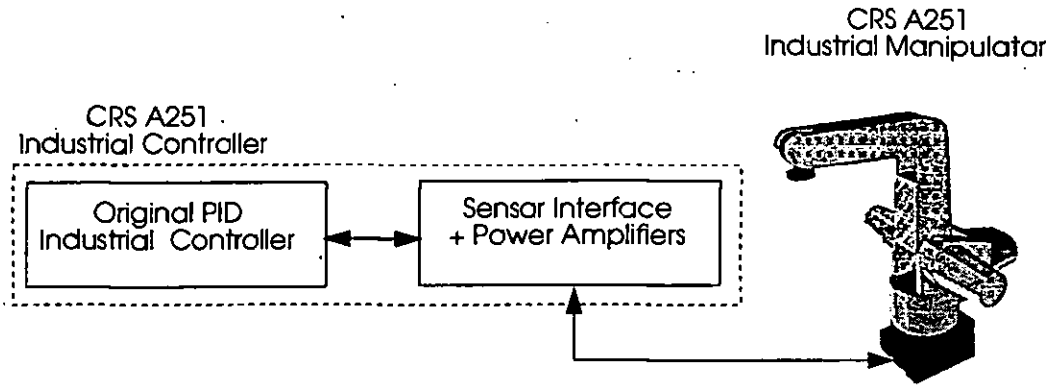

Figure 7.1: Schematic diagram of the standard CRS A251 industrial manipulator system.

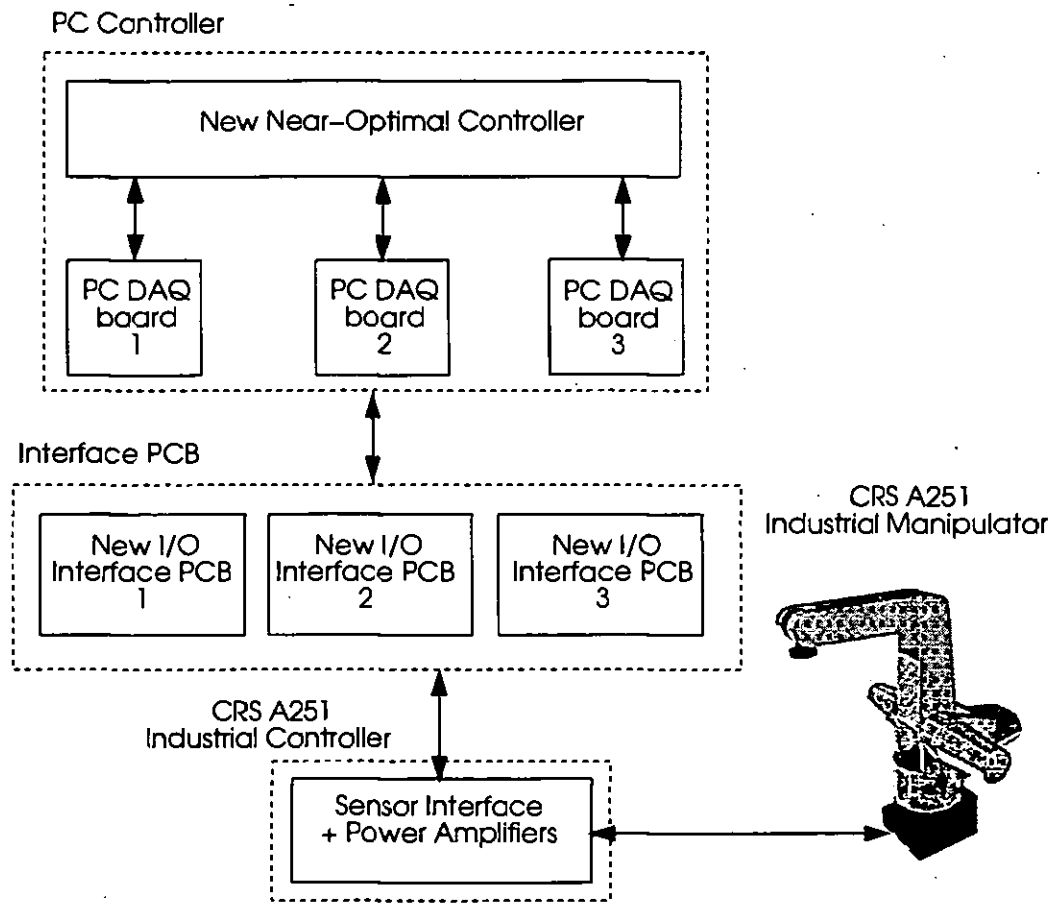

Figure 7.2: Schematic diagram of the experimental setup.

7.2.1 Current robot system controller (RSC) configuration

The characteristics of the existing RSC can be separated into three main areas as follows:

7.2.1.1 Hardware

The CRS A251 robot arm is controlled by the CRS RSC-MIA controller motherboard, in turn driven by the Intel 8087/NEC V30 16 bit microprocessor and the Intel 8087-2 math co-processor, running at a master clock frequency of 7.33 MHz. The controller box also houses standard electronic components such as the power supply, AC conditioning circuitry and motor power amplifiers. The majority of the information necessary to modify the controller circuitry was obtained from the manual [1], to which the reader is referred to for further detail, although some proprietary information had to be requested from the manufacturer in Canada.

The motherboard has 8 axis slots, five of which are used by independent PID axis cards in the standard CRS A200 Series robot, such as the CRS A251. One of the extra 3 axes available was employed to control a servo gripper. The PID axis cards are self-contained micro-controller based servo controllers [1], thus providing a fully distributed control architecture. A single Intel 8095H micro-controller integrated circuit (IC) is employed to provide a high performance PID servo loop with 1 ms closure time. The position feedback obtained from the optical encoder rotor attached to each motor shaft is updated at the same 1 ms interval. The axis card generates a command voltage output in the range of ± 10 V with 12 bits precision. This gives a voltage resolution of 0.0048 V. This signal is then shaped in the DC amplifier module, with a gain of 2.0, to the motor voltage of ± 20 V at 2 A.

A pulse train for the clockwise (CW) motion, and another for the anti-CW motion of the servo motor are available for measurement. These are obtained from the incremental optical encoder square wave feedback signals (channel A, channel B and zero crossing index) which are converted by some existing circuitry to a single pulse train on each direction. This avoids the need of dealing with the (more numerous and complicated) encoder signals, hence simplifying the sensor interface circuitry design as shown later in Section 7.2.2. The signal at this test point has a low-true pulse width of 1.8 μs (± 30 %), which sets the requirement for fairly high speed circuitry, and a period depending on the speed of the servo motor.

The controller is also fitted with motor circuit breakers which, upon detecting an overload in the system, such as a crash, energise the fail-safe electromagnetic arm breaks which cut power to the motors and hold the arm in position. They also come into play when arm power is turned off (for instance, via pressing an external Emergency-Stop button).

7.2.1.2 Software and communications

The CRS A200 Series Small Industrial Robot is a completely free-standing robot system. However, it requires a PC subsystem for running the ROBCOMM-II communication package, a programming environment which permits back-up of programs, locations and variables from the robot system to the PC. It also includes a terminal emulator for direct ASCII communication to the robot's RAPL-II operating system via a dual RS-232 link. RAPL-II is the proprietary language used by the CRS Plus family of industrial robots. It is an automation-oriented, line-structured language, designed to facilitate robot systems applications. It uses high-level commands, such as `SPEED` or `MOVE JOINT 3`, to provide a friendly interface to the operator, much along the same lines of Unimation's VAL programming language [2]. A teach pendant is also connected to the control unit to give movement freedom to the operator in operations such as homing the robot or manual-control positioning of each axis individually, effectively providing an alternative method to communicate with the robot controller.

7.2.1.3 Closed loop control and command generation

The A251 controller has multiple path generation modes which, following the outer/inner (master/slave) control loop configuration described in Section 3.2, can be summarised as 4 major modes of path control around the 1 ms joint PID closed loop algorithm [1]:

1. *Joint Interpolated.* In this mode of operation the motherboard 8086/8087 tandem processors generate command updates to the axis cards at 3.6 ms intervals which guarantee that all joints

[1] See Section 3.4 for a description of servo control mechanisms.

start and stop together. By default, the velocity of each joint varies according to a smooth spline velocity profile, although a faster but less smooth trapezoidal velocity profile is also possible.

2. *Straight Line.* This mode of control keeps the tool center point (TCP) at the tip of the arm moving in a straight line path in Cartesian space, in which command updates are performed at the slower rate of 16 ms. This mode of operation is a specialised form of the path control described below where the "knots" are calculated by RAPL-II based on a linear interpolation between end-points.

3. *Continuous Path.* This strategy defines a path curve through Cartesian space by selecting a number of intermediate points through which the robot TCP will move. The continuous path algorithm utilises a cubic spline technique (see Section 2.7) which will join the path "knot" points, in such a way as to make the velocity of the joints adjust smoothly between paths. The result of this is a motion which stops only at the end of the path. The time base for this kind of command generation is not specifically provided, but is between 16 and 40 ms.

4. *Via Path.* This path command defines a path curve through space based upon a set of intermediate points that will be approximated, but the robot may not necessarily move through them. This type of motion permits the robot to execute a series of consecutive motion commands, with the advantage that these can be entered in sequence first, and then executed continuously. Depending upon a flag set by the user, the path between points may be generated as a straight line, in which case the path commands are generated at 32 ms interval. Otherwise, 16 ms intervals are required for joint interpolated profiling.

Joint interpolation is the fastest and most "natural" way of operation of the robot controller, and subsequent comparative studies of the existing controller against the optimal controller will be run under this mode of operation.

7.2.2 New robot control configuration

As the initial test-bed for the new controller, a low cost PC-based subsystem was chosen in this work for its implementation. While a number of off-the-shelf PC-based data acquisition (DAQ) cards were employed in the general setup, there was still a need to design some in-house interface PCBs for Digital/Analog input/output (I/O) signal conditioning, as will be readily apparent below.

7.2.2.1 PC subsystem

The computer system employed in the experiments was a Pentium 75 MHz PC with a National Instruments LAB-PC+ DAQ board for each axis to be controlled, as shown in Figure 7.2. The LAB-PC+ card [3] is a low-cost multifunction analogue, digital and timing I/O card for the PC. The general configuration of the board is as follows:

- 8 analogue inputs (single-ended, or 4 differential channels) with 12-bit successive approximation analogue-to-digital converter (ADC). Bipolar (±5 V) or unipolar (0-10 V) analogue input ranges.
- 2 12-bit DACs, with bipolar or unipolar analogue output ranges.
- 24 lines of transistor-transistor logic (TTL) compatible digital I/O, also configurable by SW as 3 byte-length digital ports.
- 6 16-bit counter-timer channels for timing I/O.
- Maximum recommended analogue data acquisition rate of 83.3 kHz (for 12-bit DAC accuracy).
- Maximum built-in clock source for digital input of 2 MHz.

Bipolar settings were chosen for both input and output ports, while positional information from each joint was fed back as a 2-byte word to 2 of the digital ports, while the third was employed for digital I/O single line control data. This layout will become readily apparent when the design of the interface PCB is detailed in the following Section.

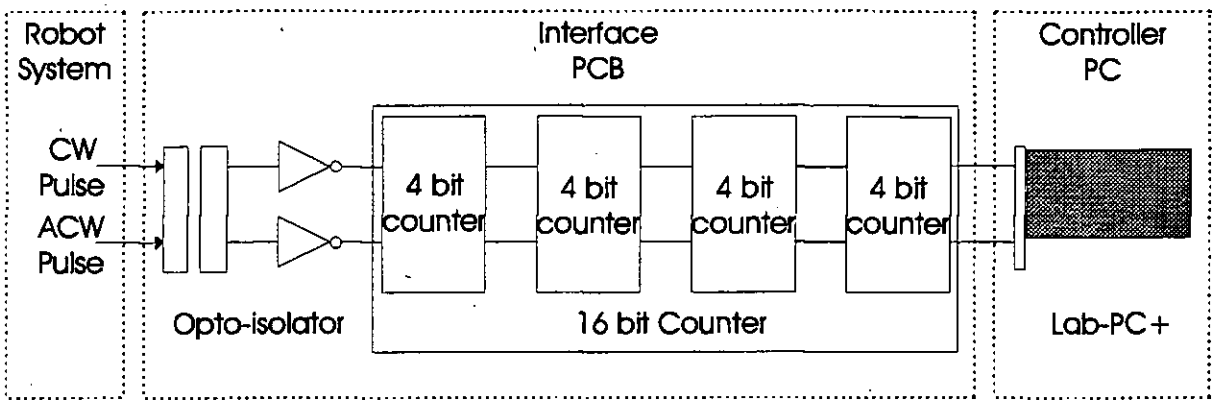

Figure 7.3: Functional block diagram of the PCB feedback stage.

The card comes with a variety of language interfaces, such as BASIC, PASCAL and C/C++, so that the programming of the board can be developed in a high-level language. DOS ANSI C in Borland's Turbo C++ environment was chosen for portability, since this allowed code written for the TELEGRIP graphical simulation environment (see Section 5.5.2) in ANSI C to be easily translated to real-time code for the controller. Other more "user friendly" programming environments, such as National Intrument's LabVIEW and LabWindows/CVI, both under Windows, were employed for code development, debugging and testing, and also for data collection. But the critical on-line controller was designed under DOS for a speedier response.

7.2.2.2 PCB subsystem

The need to develop an interface between the new PC controller subsystem and the existing robot controller has already been briefly outlined, but is now given special mention. The two main reasons can be summarised as follows:

1. Since the output range from the Lab-PC+ card is ± 5 V, an amplification stage was required to achieve the ± 10 V in the motor driving signal required by the DC linear amplifiers.

2. Given the positional output as a CW and Anti-CW pulse train, some circuitry had to be devised to obtain the state feedback information needed by the controller.

The solutions to these two design issues, presented next, were implemented on a PCB to provide a smaller size and more reliable circuitry than the traditional bread boards used during development and testing. The overall circuit diagram and the PCB artworks are included in Appendix C.

The system was designed to be modular, with the PCBs being fabricated to fit in a standard Eurocard rack. This approach permitted quick development of the prototype system, together with the added flexibility of using replaceable modules. Furthermore, a Universal Right Angle 50 way plug, which is the standard I/O Lab-PC+ connector, was also attached to the PCBs for straight off-the-shelf connection to the DAQ boards.

Output stage A common operational amplifier ($\mu A741$) in non-inverting configuration was employed for this purpose. The values of the resistor components were chosen of equal magnitude, thus achieving an amplification gain of 2.0 as desired. A potentiometer was also incorporated in the circuitry in order to eliminate the DC offset voltage at the output of the operational amplifier. It also enabled the gain to be accurately tuned and calibrated.

State feedback Figure 7.3 illustrates the configuration of the feedback subsystem implemented in the PCB. A dual high speed IC opto-isolator, the HCPL2630, was employed to provide decoupling between the existing controller feedback signals (CW and Anti-CW pulses), and the interface circuitry. This measure was taken to increase the safety and robustness of the interface circuitry, since the IC internal shield provides immunity against transient peaks beyond TTL limits.

The choice of an all-digital feedback interface was in part determined by the CW and Anti-CW TTL pulse trains at the positional test points, but it also enhanced processing speed and noise reduction.

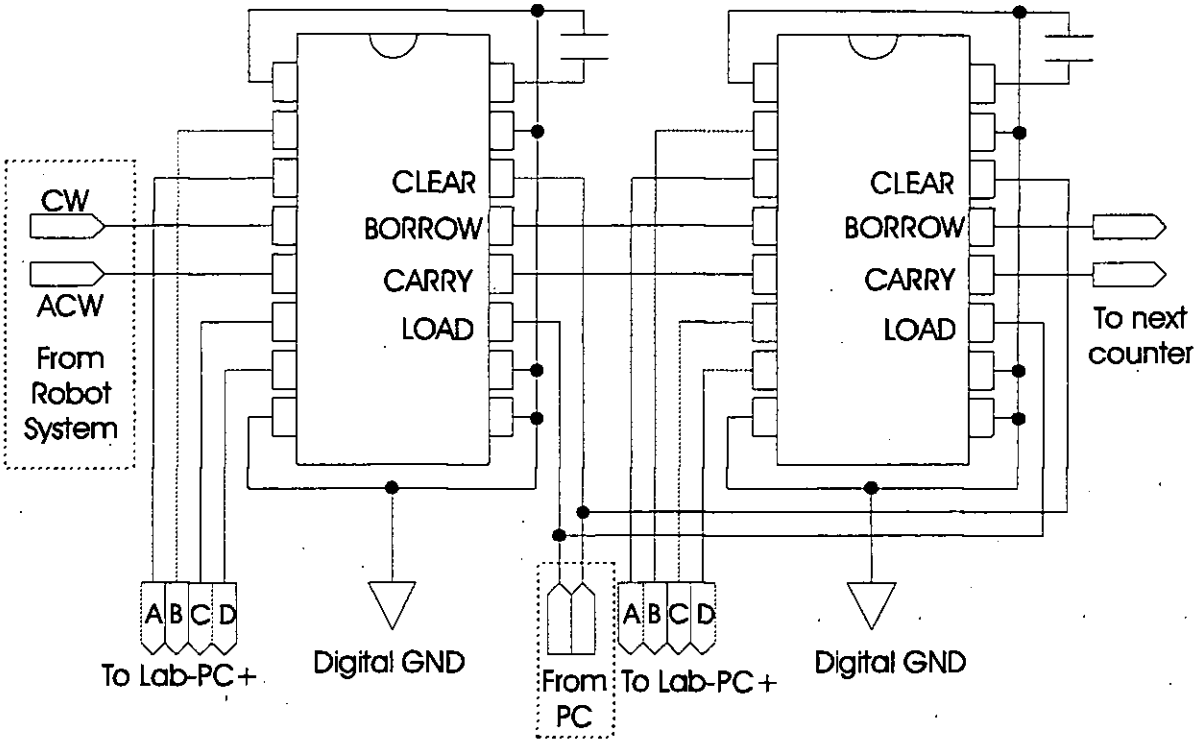

Figure 7.4: Detail of feedback counter circuitry.

The measurement of the widest range of pulses, ±34800, set by the waist joint, determined a digital word length of 16 bits, i.e., 2 of the 3 ports of the Lab-PC+ as previously stated, which was extended to all the joints for PCB design uniformity. It can be seen from the diagram that the 16 bit counter for each joint was implemented by cascading a series of 4 bit synchronous up/down binary counters, the 74LS193. The mode of operation of this IC ideally matches the CW and Anti-CW signal specifications, where the direction of counting is determined by which count input is pulsed while the other count input is held high. Moreover, the counters are designed to be cascaded without the need for external circuitry, as the detail in Figure 7.4 illustrates, where only connections for the first two counters are shown. Both `BORROW` and `CARRY` outputs are available to cascade both the up and down counting functions. The `BORROW` output produces a pulse equal in width to the count down input when the counter underflows. Similarly, the `CARRY` output produces a pulse equal in width to the count down input when an overflow condition exists. Four counters were then easily cascaded to form the desired word length of 16 bits. As is customary with IC counters, they are cyclic, so that provision for motion in the positive and negative direction from an arbitrary zero had to be taken care of by software. For further details on the hardware implementation, the reader may refer to [4].

7.3 Simulation Setup

The advantages of utilising an advanced graphical simulation engine to develop and test the novel control strategy were already introduced in Section 5.5.2. In addition to its rich graphical capabilities and robot motion modelling features, Deneb's TELEGRIP^{TM} [5] provided an open architecture which allowed proprietary algorithms to be linked directly into the motion pipeline. This was accomplished by developing user routines in ANSI C and other native programming languages to access the internal system functions and data structures used by the TELEGRIP kernel. Pre-existing system functions could also be replaced by identically named functions written in ANSI C, essentially allowing for the customisation of the overall robot motion pipeline simulation.

TELEGRIP runs on a UNIX environment, and a Silicon Graphics 4d/Indy workstation (100 MHz MIPS R4600PC processor) with a 24 bit XZ buffer accelerated graphics card running IRIX 5.2 was the computer platform employed.

7.4 Software Configuration

Figure 7.5 shows the data flow diagram for the near-optimal controller. Since the controller code developed for simulation and that written for on-line testing were both encoded in ANSI C, under the same programming structures, the flowchart concentrates only on the real-time implementation code to avoid duplicity. However, a similar SW configuration was employed in the simulation code. Only calls to the appropriate TELEGRIP graphical environment updates and the numerical integration routine (see Section 5.5.1) were specific to the simulation environment, while DAQ initialisation, I/O routines and real-time considerations were applied exclusively to the experimental setup.

It can be seen from the data flow diagram that the real-time controller SW, included in Appendix B, represents a straightforward implementation of the algorithm presented in Section 6.5.1. It is also apparent from the program that, given the multi-DoF characteristics of the plant being controlled, much of the code is duplicated for each individual joint of the robot manipulator. It was precisely this thought that initially led the author to think of embedded parallel processors (transputers) for the on-line implementation of the controller, aware that a single processor would he unable to meet the high processing needs of the algorithm. However, while the standard frequency at which the DOS clock triggers timing interrupts [2] was found too slow, at 18.2 Hz, for the real-time processing needs of the plant, the speed of calculation could be dramatically increased by nesting a piece of assembler code that changed the timer ticking frequency. The routines, which can be found in Appendix B at the beginning of the module `crsmain.c`, in the `higher_res()` and `nomal_res()` routines, allowed for an external clock frequency of 60 kHz. This, in turn, permitted much finer timer resolution (from the original 50 ms down to 0.01 ms) which primarily affected the accuracy and signal-to-noise ratio of the numerical feedback state calculations, an important problem already addressed in Section 5.3.4.1.

Under the new timing arithmetic, the total processing time to carry out a complete (all joints) control cycle was found to be around 2 ms (i.e., a frequency of 500 Hz). This execution time included not only the individual control processing times for each joint, hut also the time taken to read in the state and send out the voltage control action. However, the tight margin set by this value often allowed spurious readings, and the loop was closed around a safety sampling value of 4 ms (i.e., a frequency of 250 Hz). This is still a value much greater than 20 times the 10 Hz mechanical time constant of the arm links, chosen according to [6] to minimise the effect of sampling.

7.5 Experimental and Simulation Results

7.5.1 Evaluation criteria

The purpose of minimum-time control is to achieve fast motion along a given path, or between set-points if no path exists. It was therefore only natural that the main evaluation criterion in the experiments undertaken was settling time. A common allowable tolerance was established to define the region in the phase plane where the motion of a joint could he regarded as complete.

Definition 7.1 *A manipulator joint is considered at rest after completing a motion when the joint has reached the steady-state position (with a sensitivity of* 2.0E-02 *rad), and the velocity of the joint satisfies the inequality:*

$$|x_2| \leq 8.7\text{E-}03 \; rad/s \tag{7.1}$$

The time to reach this phase point from the beginning of the motion is represented as t_{ss}.

Furthermore,

$$t_{total} = \max_{1 \leq i \leq n} t_{ss_i} \tag{7.2}$$

represents the slowest of all joints in performing the combined motion, i.e., the time taken by the manipulator to come to rest after completion of the motion.

It is important to emphasise that these conditions set a more restrictive estimate than those reported hy other researchers. In Kahn and Roth [7], the deviations of the transformed joint angles were constrained to be within I % of their *initial* values, while the transformed joint velocities were restricted

[2] Hence allowing the user to do time-keeping tasks, such as enforcing synchronous sampling intervals.

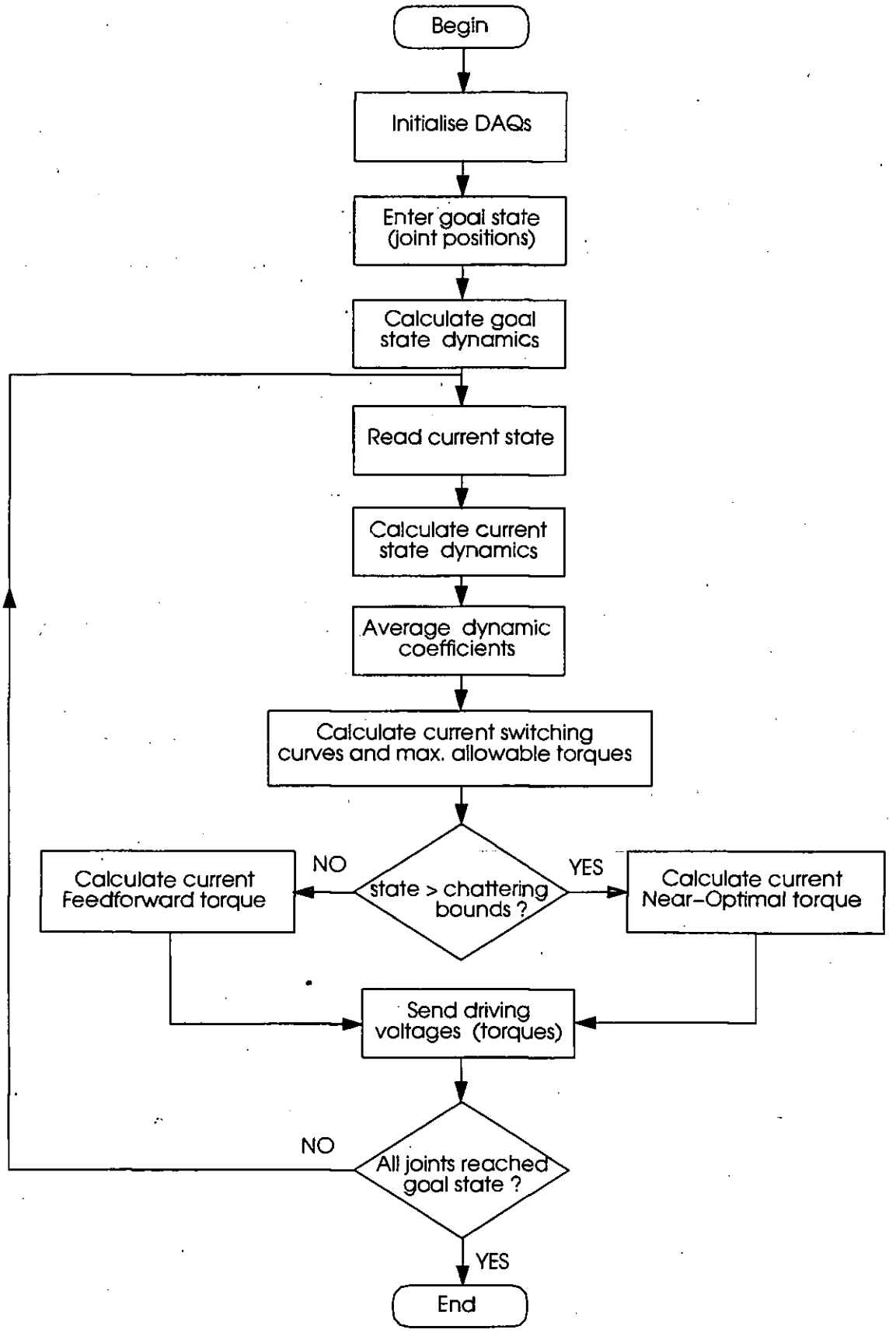

Figure 7.5: Data flow diagram of near-optimal controller.

to be less than 1 % of their maximum values. However, because these criteria were satisfied by transformed (linearised and decoupled) states, the actual joint velocities would have deviated largely from zero upon reaching the response-time. In order to compare results with the previous work, Kao *et al* [8] defined the rise-time as the time at which the deviation in all the joint angles first reached to within 1 % of their initial values. Although the conditions imposed by Definition 7.1 give way to a longer settling time than that of the stand-alone near-optimal controller [3] by taking into account the proposed model-based steady-state control too, it was felt a more appropriate measure of settling time, straight after which a new motion can be executed.

Despite focusing on timing characteristics as the key performance parameter, investigations were also conducted into other measures of performance, which included:

1. A measure of the utilisation of the torque range was also proposed as a measure of fast motion. The torque usage parameter, subsequently denoted u_{util}, was computed as the percentage of the maximum torque bound U (considered fixed here for clarity) applied throughout the motion:

$$u_{util} = \frac{100}{U}\sqrt{\sum_{t=1}^{N}\frac{u_t^2 \Delta t}{t_{ss}}} \tag{7.3}$$

   where N is the number of data points, and Δt is the closed loop sampling value, whose choice was addressed in the last Section. As can be seen from (7.3), u_{util} satisfies $0 \leq u_{util} \leq 1$, and achieves its maximum value when the motion displays a bang-bang characteristic, i.e., the actuator torque is always at the limits, $\pm U$.

2. The classical manipulator steady-state error, denoted hereafter as $Err_{ss}(rad)$, was also considered.

3. The maximum (percent) overshoot over the steady-state value, M_p.

4. One of the key aspects in the evaluation of the controller was to characterise the relative performance of the near-optimal-time controller with that of the existing PID robot controller. Hence, the potential of the proposed strategy as an alternative for industrial manipulators could be assessed. A straightforward measure of the utility of the two controllers was devised in the following speed-up percentage:

$$speed - up = 100\left(1 - \frac{t_{total}[optimal]}{t_{total}[PID]}\right) \tag{7.4}$$

   which, in a simple format, represents the (percentage) time gained ($speed - up \geq 0$) or lost ($speed - up < 0$) by employing the proposed optimal controller over the fitted PID controller, upon reaching the same destination in the workspace.

7.5.2 Illustrative examples

In order to test the efficacy of the near-minimum-time algorithm, numerical simulations and on-line experiments were performed for the set of robot configurations collected in Table 7.1. These sets of initial and final states cover a wide range of the manipulator workspace, hence giving a realistic overview of the practical performance of the controller. As can be seen in Table 7.1, the same configurations were tested in both directions, further contributing to the critical analysis of the algorithm. All examples are rest-to-rest motions.

As a basis for comparison, the same configurations of Table 7.1 were presented to both controllers, near-optimal and industrial PID, in order to obtain the performance parameters described in the previous Section. Given the vast amount of information collected from the experiments, a full presentation, which includes graphical results from the on-line experiments, numerical simulations, fitted PID controller experiments and validation data, will be restricted to two of the cases in Table 7.1, A and B. For the remaining cases, the key parameters from the real time near-optimal and standard PID controller experiments will be provided in tabular form.

[3] Comparable to the response-time defined in [7] or rise-time in [8]

Case	$x_1(t_i)(rad)$	$\Rightarrow$	$x_1(t_f)(rad)$
A	{ 0.0, -0.785, -0.61 }	$\rightarrow$	{ 0.6, -1.57, 0.0 }
B	{ 0.6, -1.57, 0.0 }	$\rightarrow$	{ 0.0, -0.785, -0.6 }
C	{ 0.0, -1.57, -0.5 }	$\rightarrow$	{ -0.4, -1.0, 0.3 }
D	{ -0.4, -1.0, 0.3 }	$\rightarrow$	{ 0.0, -1.57, -0.5 }
E	{ 0.0, 0.0, 0.0 }	$\rightarrow$	{ 0.0, -1.0, 0.0 }
F	{ 0.0, -1.0, 0.0 }	$\rightarrow$	{ 0.0, 0.0, 0.0 }
G	{ -0.5, -1.57, 0.5 }	$\rightarrow$	{ 0.8, -1.57, -0.4 }
H	{ 0.8, -1.57, -0.4 }	$\rightarrow$	{ -0.5, -1.57, 0.5 }

Table 7.1: Manipulator configurations studied.

Optimal Controller							
$x_1(t_i)$	$\Rightarrow$	$x_1(t_f)$	$t_{ss}(s)$	$t_{total}(s)$	$Err_{ss}(rad)$	u_{util} (%)	M_p (%)
0.0	$\rightarrow$	0.6	0.461	0.541	0.012	71.144	1.90
-0.785	$\rightarrow$	-1.57	0.541		0.001	91.357	0.43
-0.61	$\rightarrow$	0.0	0.365		0.019	63.933	3.01

PID Controller							
$x_1(t_i)$	$\Rightarrow$	$x_1(t_f)$	$t_{ss}(s)$	$t_{total}(s)$	$Err_{ss}(rad)$	u_{util} (%)	M_p (%)
0.0	$\rightarrow$	0.6	0.552	0.596	0.0	54.076	0.69
-0.785	$\rightarrow$	-1.57	0.596		0.001	91.057	2.74
-0.61	$\rightarrow$	0.0	0.536		0.007	57.466	1.05

Table 7.2: Case A.

7.5.2.1 Case A

The full set of results obtained for Case A are graphically depicted in Figures 7.6 - 7.12. The near-minimum-time state profile for this example is shown in Figure 7.6 for measurements, and Figure 7.7 for simulation. In both instances, the left plot shows the joint displacements, while the right plot shows the joint velocities. The axes are scaled in link radians versus time. As the graphs show, experimental and simulated curves agree satisfactorily.

It can also be seen from the plots that the upper-arm, Joint 2, is the controlling DoF in this case, i.e., the slowest joint. This result is also numerically collected in Table 7.2, top, where it is clear that the optimal settling time, $t_{total} = 0.541$ s, corresponds to the time taken by Joint 2 to reach the goal point, as given by Equation (7.2). This value compares favourably to the timing performance of the PID controller in executing the same motion, $t_{total} = 0.596s$, as shown in Table 7.2, bottom. The trajectory followed under this strategy is graphically shown in Figure 7.9. It is easy to observe how the traditional scheme of geometric trajectory planner coupled with error-driven PID control produces a more co-ordinated trajectory than that of the near-minimum-time optimal controller of Figure 7.6,

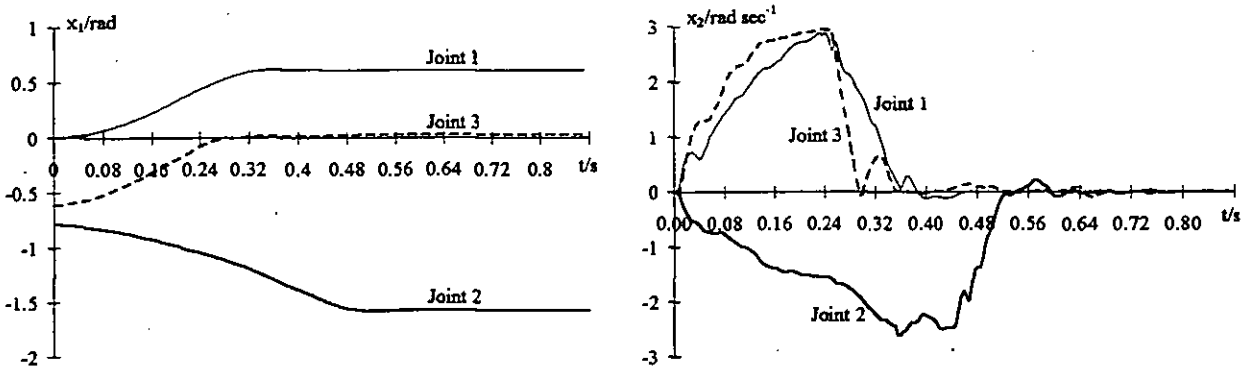

Figure 7.6: Case A. Measurements of joint states under optimal control.

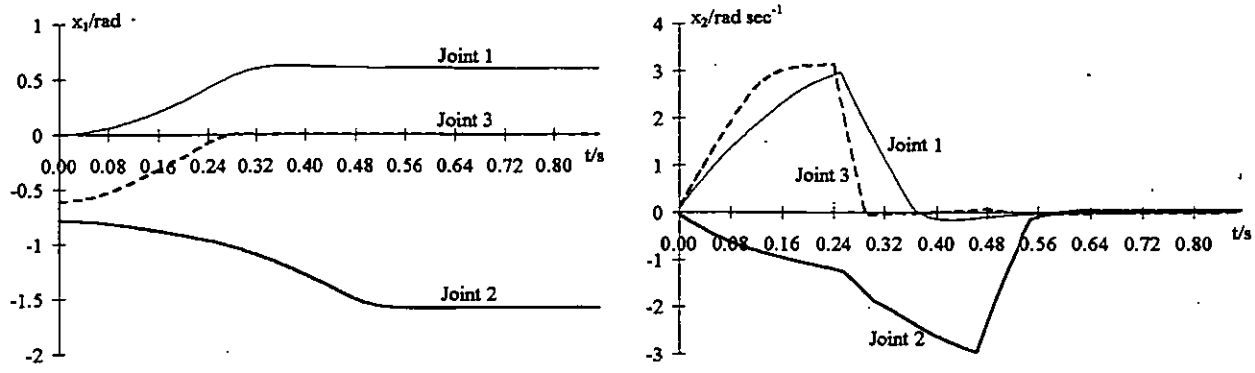

Figure 7.7: Case A. Simulation of joint states under optimal control.

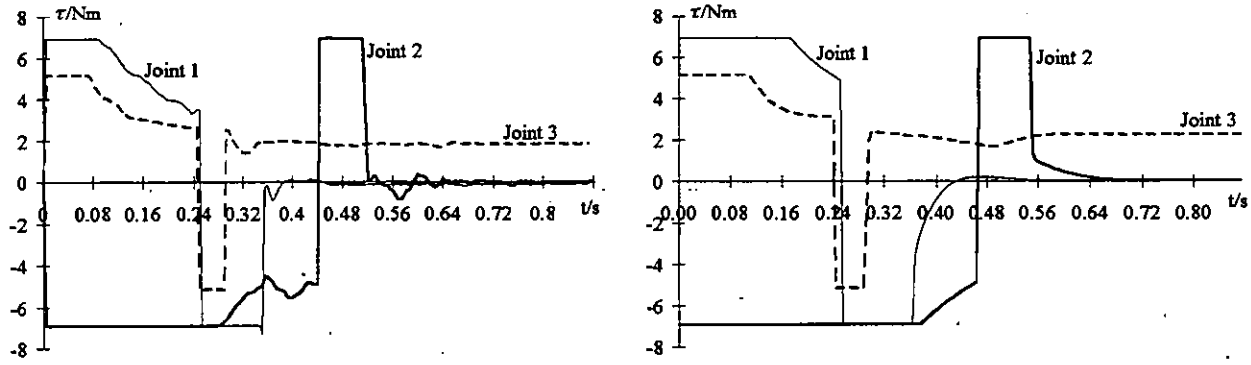

Figure 7.8: Case A. Measurements (left) and simulation of optimal control torques.

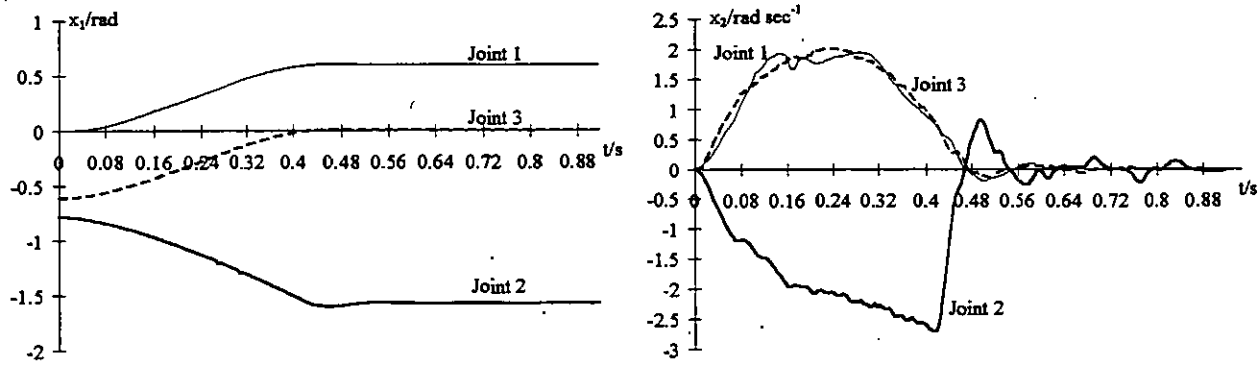

Figure 7.9: Case A. Measurements of joint states under PID control.

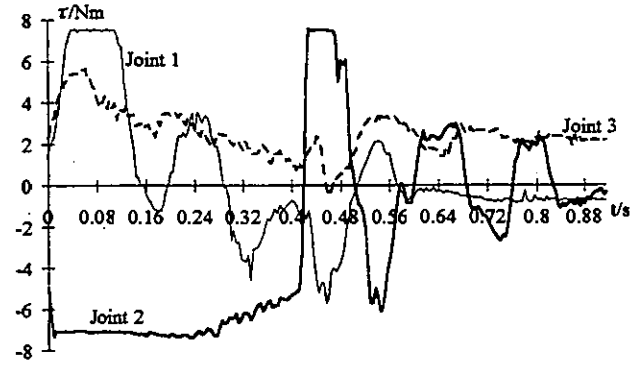

Figure 7.10: Case A. Measurements of joint torques under PID control.

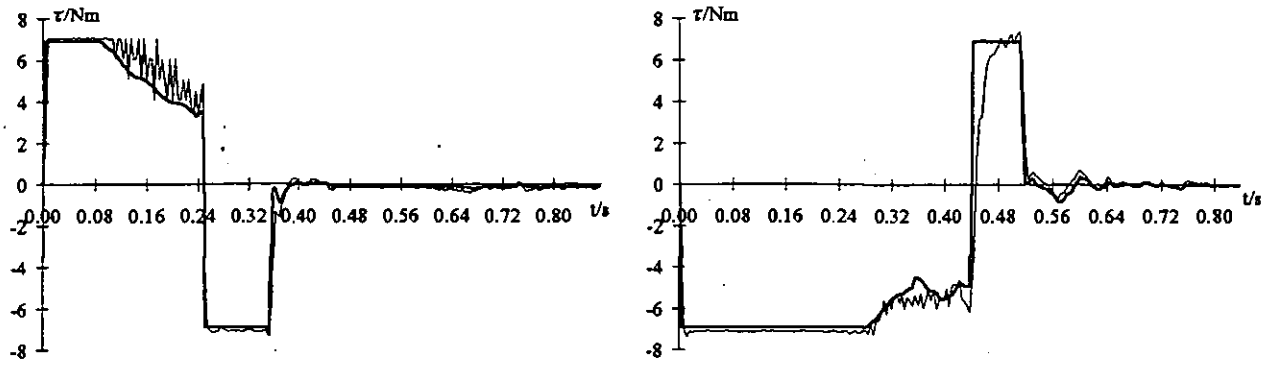

Figure 7.11: Case A. Demanded (bold) and measured actuator optimal driving torques. Joint 1 (left) and 2.

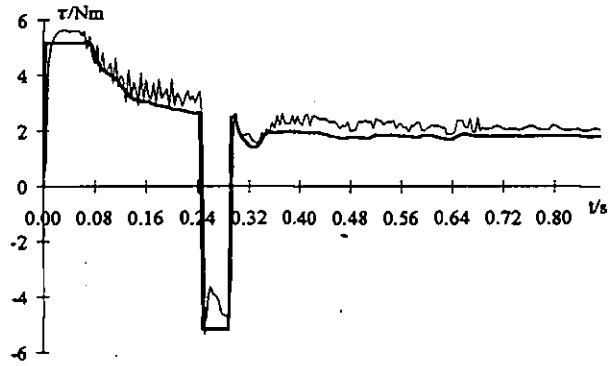

Figure 7.12: Case A. Demanded (bold) and measured actuator optimal driving torques. Joint 3.

although there is still one joint that exhibits the limiting characteristics. In general, that would be the same joint which imposes its limitations in the near-optimal strategy, although the steady-state oscillations might render the controlling joint to be different from the limiting one, as will be the case for run B, analysed in the following Section.

The close match between theoretical (right) and experimental (left) results obtained for the near-optimal torque profile shown in Figure 7.8 further contributes to the validation of the dynamic modelling and simulation environment employed. A comparison of these results and those depicted in Figure 7.10 clearly reveals the general characteristic of a saturating joint as a limiting condition in both strategies. It is clear from these graphs that, while the upper-arm is driven close to its maximum capabilities under both strategies, the other joints are correspondingly adjusted under traditional geometric trajectory planning and PID control (Figure 7.10) to allow for a fairly simultaneous motion of all the joints. This is reflected back in the lower percentage of torque utilisation for each joint, u_{util}, in comparison with the near-optimal controller utilisation values shown in Table 7.2.

A peculiarity of this case shows how the high-speed demands of the industrial controller have forced the utilisation of the limiting joint to be relatively close to its corresponding near-optimal u_{util}, 91.057 % versus 91.357 % respectively. However, this is not necessarily a general characteristic of the scheme, as results from other experiments reviewed in later Sections suggest (e.g. Case B). In fact, it is rarely so. On the other hand, the traditional voltage-driven PID scheme does indeed show a tendency to introduce more torque oscillations near the steady-state than the near-optimal controller when driven at high speeds, a direct consequence of its (purely) positional error-driven characteristic and lack of direct torque control [4]. While this effect, clearly visible in Figure 7.10, does not significantly increase the %age overshoot (around 3 % in hoth cases), it does, nevertheless, augment settling time due to the appreciable velocity oscillations caused near the steady-state (as seen in Figure 7.9, right). This can be more effectively dealt with under the near-optimal controller strategy for two intrinsic reasons:

[4] For further details on this design feature the reader may refer to Section 5.4, where the problem and its solution were introduced.

Optimal Controller							
$x_1(t_i)$	$\Rightarrow$	$x_1(t_f)$	$t_{ss}(s)$	$t_{total}(s)$	$Err_{ss}(rad)$	u_{util} (%)	M_p (%)
0.6	$\rightarrow$	0.0	0.449	0.449	0.004	77.187	1.16
-1.57	$\rightarrow$	-0.785	0.409		0.002	84.122	0.23
0.0	$\rightarrow$	-0.6	0.273		0.001	68.283	0.16

PID Controller							
$x_1(t_i)$	$\Rightarrow$	$x_1(t_f)$	$t_{ss}(s)$	$t_{total}(s)$	$Err_{ss}(rad)$	u_{util} (%)	M_p (%)
0.6	$\rightarrow$	0.0	0.568	0.588	0.0	54.778	0.70
-1.57	$\rightarrow$	-0.785	0.588		0.001	61.137	0.55
0.0	$\rightarrow$	-0.6	0.52		0.001	31.676	0.66

Table 7.3: Case B.

1. The optimal torque action is designed so as to drive the manipulator to the origin. Hence, less velocity oscillations would occur in the first place, as shown in Figure 7.6.

2. The switching to a steady-state torque control, such as the (partially error-driven) feedforward controller proposed in this work, can more judiciously damp the remaining torque oscillations, as clearly depicted in Figure 7.8.

It is also interesting to note the reduced maximum steady-state errors achieved under both strategies. While the PID integral action managed to effectively reduce this error to less than 0.007 rad [5], the near-optimal plus feedforward controller was not able to decrease this value to less than 0.019 rad. Since the feedforward controller comes into action for a small period of time near the end of the motion, where errors are kept small and building up to small values for the foregoing reasons, the introduction of an integral action showed no significant improvements, and was opted out due to its associated potential problems of stability (see Section 3.5 for more details on the strategy). Despite this result, the "gross motion" characteristic for which the algorithm was originally intended should he kept in mind at this point. Under such a perspective, time is assigned a higher relevance than other parameters, and the steady-state accuracy achieved should still be regarded as an exceptional outcome of the proposed strategy.

Figures 7.11 and 7.12 depict the commanded actuator torque which results from the near-optimal controller, and the actual control torque exerted in response, by the three major manipulator joints under analysis. These results are included here to extend the validation of the approach to accurately command the actuator torques, described in Section 5.4.3, under extreme conditions such as those of the near-optimal controller. It can be seen how, other than minimal errors from overlooking some electro-mechanical dynamic effects (and already considered in Section 5.4.3), the modelling accurately reflects the actuating subsystem dynamics.

7.5.2.2 Case B

The full set of results obtained for run B are graphically collected in Figures 7.13 to 7.19. Following the structure previously adopted for Case A, the near-minimum-time state profiles for measurement and simulation are shown side by side in Figures 7.13 and 7.14 respectively. These satisfactorily reflect how the general character of the solution can be accurately reproduced in the modelling environment. An important feature of these curves corresponds to the velocity spikes measured on the experiments, which are not present in the simulations. The reason behind this variation in behaviour between experimental and simulation results can be mainly attributed to noise bursts in the measurement of position, and the consequent amplification introduced by its numerical differentiation carried out to obtain a full state feedback. The reasons for employing this approach in the work presented here, and possible alternatives for future implementations, can be found in Section 5.3.4.1 where the problem was first envisaged. Also, the fact that a DOS PC is not a real-time operating system environment would also mean that a "variable" synchronous sampling time might have occurred when the microprocessor

[5] Which might still be regarded as slightly larger than expected, although this is probably explained by the extremely high velocities demanded from the industrial controller

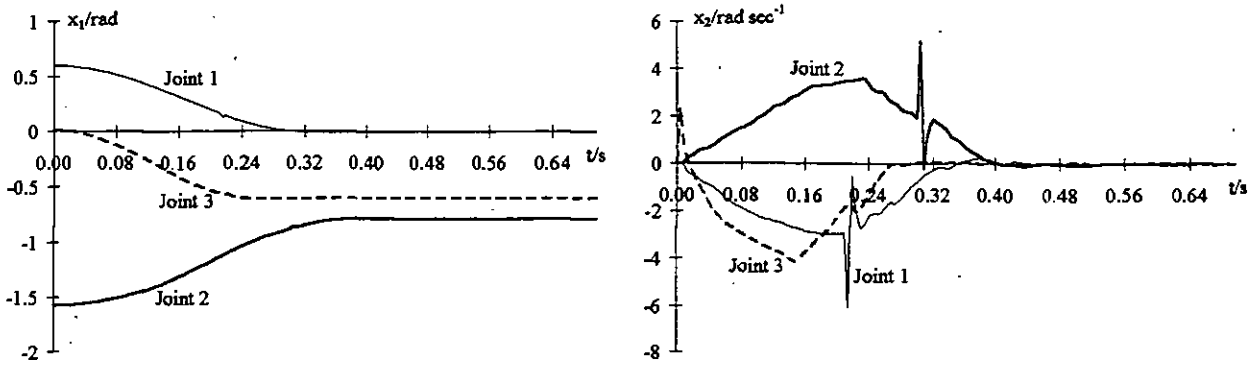

Figure 7.13: Case B. Measurements of joint states under optimal control.

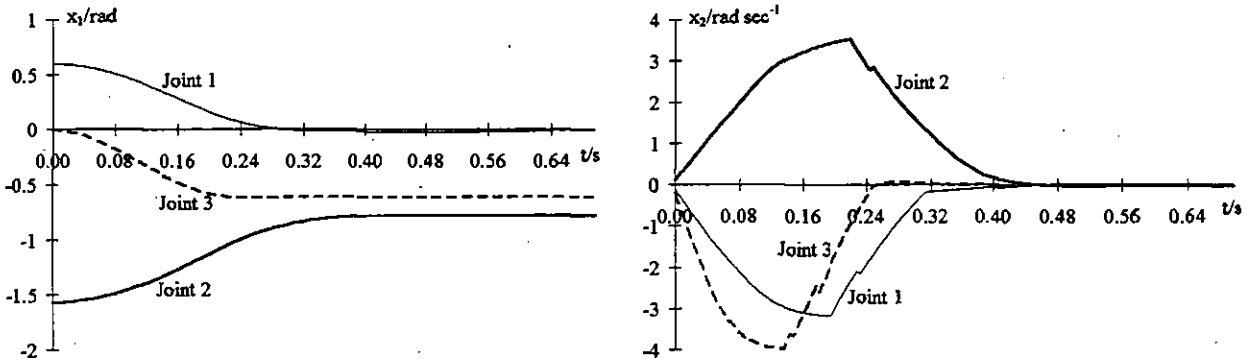

Figure 7.14: Case B. Simulation of joint states under optimal control.

could not have served the clock interrupt as requested by the controlling program, due to its low priority against other system interrupts. This would have also shown spikes in the velocity measures. Furthermore, it sould not be underestimated the quick and large switching in the motor voltage, and the consequent electromagnetic forces generated, which might have also contributed to the bursty velocity readings.

It is important to realise the "corrective" action injected straight-away by the optimal controller in order to counteract this deviation of the manipulator from the ideal trajectory profile. This is depicted as sudden and short-lived torque changes in Figure 7.15. However, it should be emphasised that while some of these sudden bursts correspond to noisy readings to which the controller adapts (easily identifiable by a comparison of superimposed jumps between measured velocity and torque curves), others should be attributed to the implicit errors in estimating the manipulator dynamics for the remaining motion, mathematically represented by the λ parameter in Equation 6.27. Although the importance of the parameter is analysed in depth in Section 7.5.2.4, it is nevertheless obvious at this stage how the closer the chosen coefficient should efficiently represent the variable manipulator dynamics over time, the fewer control switches would be required to compensate for "unforeseen" changing dynamics along the sliding trajectory [6]. In other words, less short "compensations" will be required from the controller to keep the manipulator at the desired optimal state trajectory. This distinction is clearly visible from a comparison of the simulated and experimental near-optimal torque profiles of Figure 7.15. It should nevertheless be emphasised that, irrespective of the nature of the short corrective actions, the result is always one of smooth displacements of the robot links, as the positional graphs clearly illustrate.

Case B also demonstrates another important feature of the torque-based near-optimal controller in comparison to the error-based PID controller - that is the better use of the available resources when motion of the links takes place predominantly with gravity. This is illustrated when Figure 7.15 and 7.17 are compared, and also by the results collected in Table 7.3. Here, PID timing shows a similar figure to that of Case A, 0.588 s, while t_{total} for the near-optimal controller is decreased down to 0.449 s.

[6] Updated at each sample time.

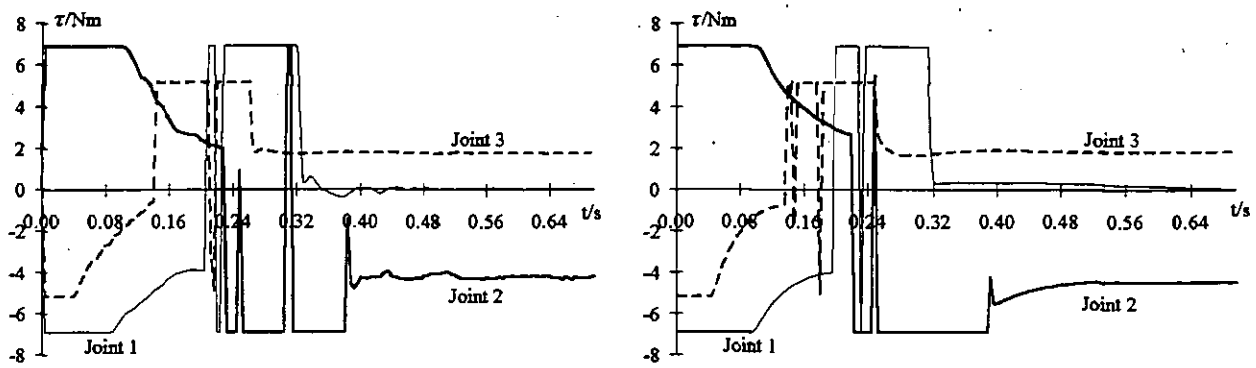

Figure 7.15: Case B. Measurements (left) and simulation of optimal control torques.

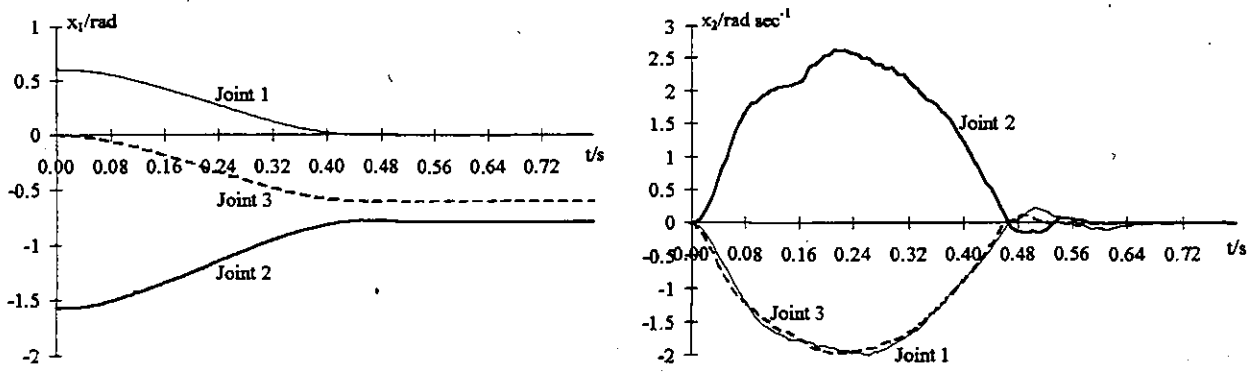

Figure 7.16: Case B. Measurements of joint states under PID control.

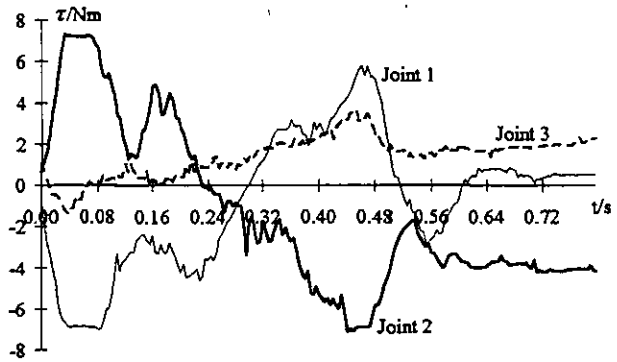

Figure 7.17: Case B. Measurements of joint torques under PID control.

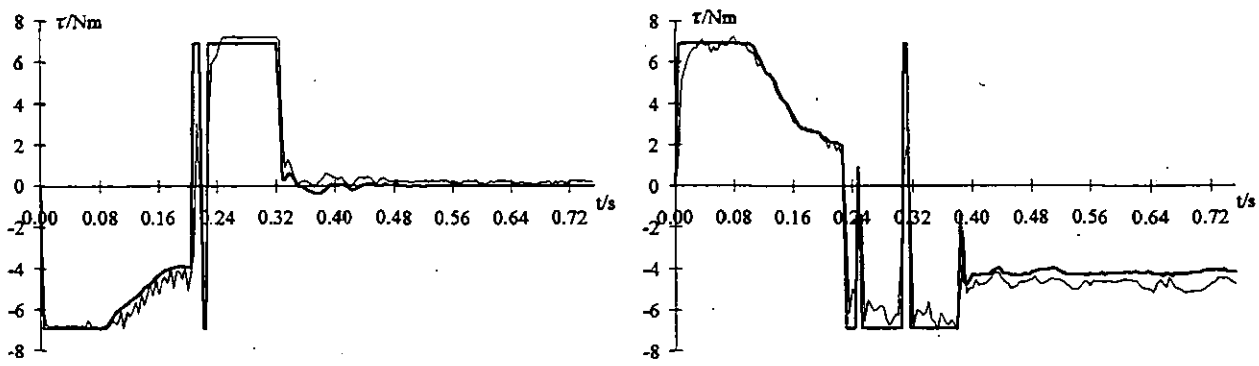

Figure 7.18: Case B. Demanded (bold) and measured actuator optimal driving torques. Joint 1 (left) and 2.

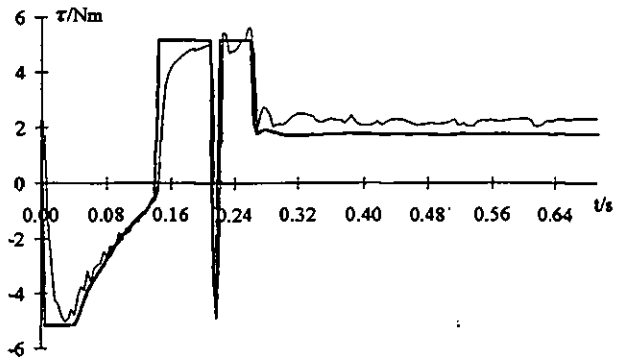

Figure 7.19: Case B. Demanded (bold) and measured actuator optimal driving torques. Joint 3.

Optimal Controller							
$x_1(t_i)$	$\Rightarrow$	$x_1(t_f)$	$t_{ss}(s)$	$t_{total}(s)$	$Err_{ss}(rad)$	u_{util} (%)	M_p (%)
0.0	$\rightarrow$	-0.4	0.465	0.493	0.004	69.711	0.81
-1.57	$\rightarrow$	-1.0	0.433		0.01	83.064	0.18
-0.5	$\rightarrow$	0.3	0.493		0.004	76.34	0.77

PID Controller							
$x_1(t_i)$	$\Rightarrow$	$x_1(t_f)$	$t_{ss}(s)$	$t_{total}(s)$	$Err_{ss}(rad)$	u_{util} (%)	M_p (%)
0.0	$\rightarrow$	-0.4	0.472	0.528	0.005	39.012	0.49
-1.57	$\rightarrow$	-1.0	0.524		0.0	48.663	0.39
-0.5	$\rightarrow$	0.3	0.528		0.0	61.210	0.35

Table 7.4: Case C.

Figure 7.15 indicates how, as in Case A, the available torque for each joint is maximised under near-optimal control. However, since gravity need not now be overcome, the full torque can be applied in accelerating the body, hence reaching the goal-point faster. This is not the case for the PID controller, which due to its nature can hardly take advantage of this fact. Working with gravity surely facilitates the objective of the controller in keeping the manipulator closer to the desired trajectory, thus smaller errors are generated along the trajectory. Consequently, torque requirements along the trajectories are kept lower than in Case A, as Figure 7.17 and the u_{util} parameter of Table 7.3 illustrates. Essentially, only at the beginning of the motion, when the error is larger, a larger control action burst is required to get the manipulator moving fast. For the rest of the motion, much of the available capabilities of the actuators are not fully employed.

Results listed in Table 7.3 also show favourable steady-state error (less than 0.004 rad) and %age overshoot (less than 1.16 %) values to those of Case A, and fairly similar to the equivalent parameters under PID control (less than 0.001 rad and 0.70 % respectively). In that respect, the same issues previously raised for Case A about the "gross motion" nature of the algorithm can be noted now. As with Case A, Figures 7.18 and 7.19 show, in bold, the actuator torques commanded by the near-optimal controller for the three joints individually [7], the graphs being superimposed by the actual control torques exerted in response. Yet again, a close match is obtained, which illustrates the accuracy of the actuator dynamics modelling. As previously pin-pointed in Chapter 5, there is a slightly poorer matching between model and measurements of joint 3, with slower rise and settling times as shown in Figure 7.19. The reader is referred back to Section 5.6 where this fact was initially addressed.

7.5.2.3 Other cases

The parameter values for the remaining cases analysed are collected between Tables 7.4 and 7.9, while the overall near-optimal controller/PID speed-ups are listed in Table 7.10.

In general, results indicate that the near-optimal controller performs well for a wide range of motions that cover the majority of the manipulator workspace. Speed-ups as exceptional as 24.4 % (Case

[7] Collectively shown in Figure 7.15, left.

Optimal Controller							
$x_1(t_i)$	$\Rightarrow$	$x_1(t_f)$	$t_{ss}(s)$	$t_{total}(s)$	$Err_{ss}(rad)$	u_{util} (%)	M_p (%)
-0.4	$\rightarrow$	0.0	0.497	0.517	0.001	87.728	1.78
-1.0	$\rightarrow$	-1.57	0.517		0.006	94.316	1.0
0.3	$\rightarrow$	-0.5	0.501		0.003	78.065	0.54

PID Controller							
$x_1(t_i)$	$\Rightarrow$	$x_1(t_f)$	$t_{ss}(s)$	$t_{total}(s)$	$Err_{ss}(rad)$	u_{util} (%)	M_p (%)
-0.4	$\rightarrow$	0.0	0.472	0.536	0.0	39.671	0.39
-1.0	$\rightarrow$	-1.57	0.536		0.002	67.851	0.63
0.3	$\rightarrow$	-0.5	0.468		0.0	38.079	0.37

Table 7.5: Case D.

Optimal Controller							
$x_1(t_i)$	$\Rightarrow$	$x_1(t_f)$	$t_{ss}(s)$	$t_{total}(s)$	$Err_{ss}(rad)$	u_{util} (%)	M_p (%)
0.0	$\rightarrow$	0.0	0.032	1.383	0.0	0.29	0.11
0.0	$\rightarrow$	-1.0	1.383		0.012	96.09	1.54
0.0	$\rightarrow$	0.0	0.056		0.0	42.928	0.61

PID Controller							
$x_1(t_i)$	$\Rightarrow$	$x_1(t_f)$	$t_{ss}(s)$	$t_{total}(s)$	$Err_{ss}(rad)$	u_{util} (%)	M_p (%)
0.0	$\rightarrow$	0.0	N/P	N/P	N/P	N/P	N/P
0.0	$\rightarrow$	-1.0	N/P		N/P	N/P	N/P
0.0	$\rightarrow$	0.0	N/P		N/P	N/P	N/P

Table 7.6: Case E (N/P = Not Possible).

Optimal Controller							
$x_1(t_i)$	$\Rightarrow$	$x_1(t_f)$	$t_{ss}(s)$	$t_{total}(s)$	$Err_{ss}(rad)$	u_{util} (%)	M_p (%)
0.0	$\rightarrow$	0.0	0.028	0.533	0.0	0.325	0.03
-1.0	$\rightarrow$	0.0	0.533		0.008	90.35	0.43
0.0	$\rightarrow$	0.0	0.357		0.007	36.762	1.46

PID Controller							
$x_1(t_i)$	$\Rightarrow$	$x_1(t_f)$	$t_{ss}(s)$	$t_{total}(s)$	$Err_{ss}(rad)$	u_{util} (%)	M_p (%)
0.0	$\rightarrow$	0.0	0.0	0.616	0.0	1.923	0.01
-1.0	$\rightarrow$	0.0	0.616		0.003	61.04	0.11
0.0	$\rightarrow$	0.0	0.076		0.0	41.145	0.1

Table 7.7: Case F.

Optimal Controller							
$x_1(t_i)$	$\Rightarrow$	$x_1(t_f)$	$t_{ss}(s)$	$t_{total}(s)$	$Err_{ss}(rad)$	u_{util} (%)	M_p (%)
-0.5	$\rightarrow$	0.8	0.521	0.521	0.001	73.318	0.04
-1.57	$\rightarrow$	-1.57	0.377		0.011	19.428	1.91
0.5	$\rightarrow$	-0.4	0.352		0.005	71.372	0.0

PID Controller							
$x_1(t_i)$	$\Rightarrow$	$x_1(t_f)$	$t_{ss}(s)$	$t_{total}(s)$	$Err_{ss}(rad)$	u_{util} (%)	M_p (%)
-0.5	$\rightarrow$	0.8	0.676	0.676	0.0	52.811	0.78
-1.57	$\rightarrow$	-1.57	0.12		0.0	11.747	0.06
0.5	$\rightarrow$	-0.4	0.656		0.0	33.181	0.26

Table 7.8: Case G.

Optimal Controller							
$x_1(t_i)$	$\Rightarrow$	$x_1(t_f)$	$t_{ss}(s)$	$t_{total}(s)$	$Err_{ss}(rad)$	u_{util} (%)	M_p (%)
0.8	$\rightarrow$	-0.5	0.513	0.541	0.003	70.668	0.42
-1.57	$\rightarrow$	-1.57	0.0		0.004	6.437	0.9
-0.4	$\rightarrow$	0.5	0.541		0.001	75.383	1.27

PID Controller							
$x_1(t_i)$	$\Rightarrow$	$x_1(t_f)$	$t_{ss}(s)$	$t_{total}(s)$	$Err_{ss}(rad)$	u_{util} (%)	M_p (%)
0.8	$\rightarrow$	-0.5	0.672	0.672	0.0	51.425	0.79
-1.57	$\rightarrow$	-1.57	0.0		0.0	7.288	0.08
-0.4	$\rightarrow$	0.5	0.596		0.0	59.023	0.29

Table 7.9: Case H.

Case	Speed up (%)
A	9.23
B	24.4
C	6.65
D	3.54
E	N/P
F	13.5
G	22.9
H	19.5
Average	14.25

Table 7.10: Near-optimal controller/PID speed-up.

B) have been achieved, 3.54 % being the lowest for Case D. On average, a value of 14.25 % has been obtained for 7 of the 8 cases considered. The 8th, Case E, is worth special mention. Here, the controller is asked to keep joints 1 and 3 stationary at the initial configuration (both at 0.0 rad), while transfering joint 2 from 0.0 to -1.0 rad, hence testing the response of the controller to keep joints locked at a certain position. It is shown in Table 7.6 how the PID controller was not able to perform the required motion at maximum speed as specified. This can be explained by the intensive torque requirements from Joint 2 in order to accelerate the manipulator from its initial configuration. The dominant gravity-induced torque at that point, jointly with the requirements of full-speed trajectory tracking, caused a following error beyond the permissible value of the controller. By design, an occurrence of this characteristic immediately triggers a safe routine that cuts power to the manipulator arm to avoid overload damage to the motors. The motion was, however, successfully completed when the speed requirements were reduced to 65 %. A different scenario took place under near-optimal control due to its torque-based characteristic. Although the overall motion was relatively slow before Joint 2 managed to counteract the static torque components and accelerate the body, it did so by safely exerting the maximum rated torque available at each instant, thus completing the motion after 1.383 s.

Other conclusions can he drawn from the results presented here, some of which were recently published in a preliminary results paper [9]:

- Continuing with the trends exhibited by Cases A and B, it can he seen how the available torque utilisation in all cases is consistently higher under the near-optimal controller, an expected performance given the design specifications. As a consequence, faster settling times are achieved. Reduced steady-state oscillations also contribute to this result, partially induced by the direct control of the torque, and also the switching to a feedforward controller near the end-state as discussed in Case A.

- It may also be seen that similar, if only slightly larger, peak values of the response curve, were measured in comparison with the PID strategy. This is a reflection of the good damping features of the combined near-optimal controller, despite the large speeds and torques under which the manipulator is driven throughout the motion.

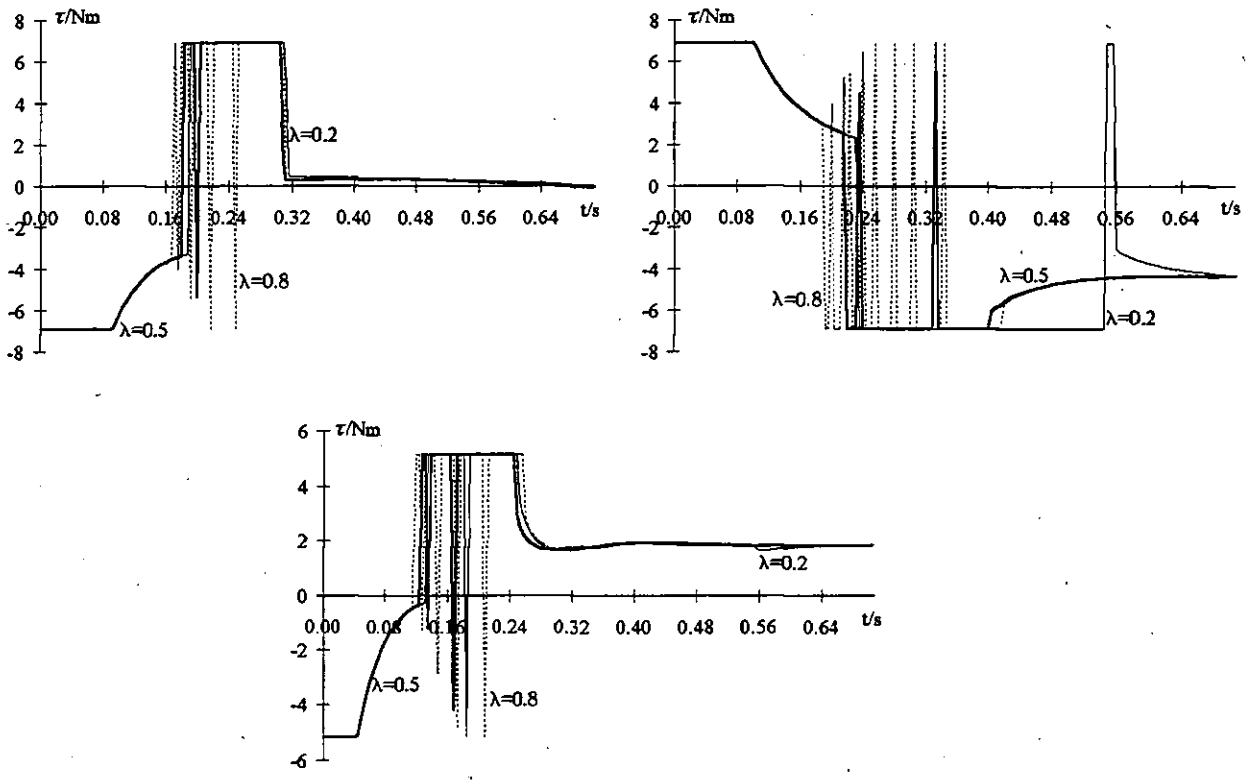

Figure 7.20: Case B. Influence of factor λ in near-optimal torque profile. Joint 1 (top left), Joint 2 (top right) and Joint 3.

- In addition to high operational speed, results from the proposed approach also highlight its ability to achieve small steady-state errors, although not as good as the PID, for the reasons exposed in Section 7.5.2.1. This is, however, a small drawback under the "gross motion" behaviour the near-optimal controller is designed for.
- Another general characteristic of the controller is the wiser use of the resources when the motion takes place with gravity, as demonstrated when Case B was analysed. This ability applies in particular to Joint 2, since this joint is designed to support the reaction torques exerted by the rest of the manipulator structure. It can be seen in Table 7.10 how the best performances are gained under these circumstances. Hence, Cases B, C, F and G achieve better speed-ups than their counterpart motions, A, D, E and H respectively. It is shown how speed-up gains between parallel cases with same end-points are considerably influenced by the gravitational effect in all cases, except between Cases G and H, which is not so affected because Joint 2 is kept stationary and only Joint 3 moves with the gravitational field.
- The combined HW and SW robot configurations has also given way to undesirable velocity peaks as a result of the differentiation process employed. This was necessary in order to obtain this parameter from the position measurements. While some alternatives have already been devised in the thesis (see Section 5.3.4.1) to avoid such a problem, results have nevertheless shown the adequate behaviour of the controller to counteract these errors, as previously explained in Section 7.5.2.2 when Case B was analysed.

7.5.2.4 Influence of parameter λ

Section 6.5 introduced the concept of the averaged dynamics as an attempt to model the variable plant dynamics, based on the limited feedback information available at each time instant. Key to the proposed methodology was the dynamic factor λ, whose intuitive importance has already been anticipated in Section 7.5.2.2, when Case B was analysed.

The effects of the variation of the parameter λ in Equation 6.27 is demonstrated by the simulated near-optimal torque profiles shown in Figure 7.20. These correspond to the response from each individual

manipulator joint when the robot was prescribed to accomplish Case B motion. While the whole range of λ coefficients from 0 to 1 was tested in steps of 0.1, only three cases, namely those corresponding to values of 0.2, 0.5, and 0.8, are presented here for the sake of clarity, the general behaviour of the remaining cases being accomodated by these three parameters.

A close look at Equation 6.27 reveals how, mathematically, large coefficients of λ correspond to an increasing importance of the end-point dynamics in the overall dynamic behaviour of the manipulator, while lower coefficients imply a preponderance of the current configuration. Which boundary configuration should he given primary consideration seems, *a priori*, a difficult and elusive question. However, in view of the experimental data collected in Figure 7.20, it can be concluded that too large values show a tendency to generate far more switchings than small values. This is particularly true when the manipulator is sliding along the optimal trajectory towards the origin, i.e., after the manipulator state trajectoy hits the switching curve . This indicates a need to "accomodate" the manipulator dynamics to its true dynamics more often throughout the motion, although it is also shown that this does not necessarily mean slower motions, given the adaptive nature of the proposed algorithm. On the other hand, a predominance of the current manipulator configuration in the averaged dynamics seems to be more prone to higher overshoots and longer settling times, as suggested by results from Joint 2. The fact that this situation does not arise in the other joints is probably explained by the dominant influence of the other joints's dynamic and static forces in Joint 2, which is not so much the case for the other two links analysed.

In view of these facts, it was concluded that a medium-to-small coefficient would avoid unnecessary dynamic "adaptions", while still giving an optimal performance with a wide dynamic range. Hence, $\lambda = 0.4$ was found most adequate, and all the experimental results shown in this Chapter (Cases A to H) have been obtained under this λ configuration.

7.6 Summary and Discussion

The simulation and implementation of the proposed novel methodology for the near-optimal control and trajectory planning of industrial manipulators has been presented in this Chapter. The required HW and SW developments which permitted the real-time control of the 5 DoF CRS A251 robotic system from a stand-alone PC have been described in full.

Further to the validation of the robotic arm dynamic model, carried out in Chapter 5, new sets of results have heen presented here which satisfactorily compare simulation and practical results, thus supporting the suitability of the manipulator dynamic model for the design of the controller.

Out of the various manipulator configurations, arbitrarily selected to represent the robot workspace as a whole, results obtained from Case B, in Section 7.5.2.2, suggested a maximum time improvement of nearly 25 % over the standard trajectory planner plus PID tracking strategy, with which the industrial manipulator was fitted. However, in practice, the average improvement for the eight show-cases analysed reflected a satisfactory average time optimisation of 14.25 %, with the lower speed-up boundary heing 3.54 % for Case D. This experimental evidence for near-time-optimal strategies as a favoured control method for unconstrained point-to-point trajectory planning and control is backed up by the maximisation of the manipulator capabilities thoughout the motion, in comparison with the standard robot controller performance.

The consideration of other evaluation criteria, such as steady-state error and overshoot, have almost consistently achieved better values under the PID industrial controller, on the other hand not an unexpected result given its design specifications. However, it is important to point out the "gross motion" characteristic for which the algorithm was originally intended. Within such a framework, time is assigned a higher relevance than other parameters, such as steady-state accuracy, and it is in that respect the near-optimal-time controller has clearly outperformed the industrial controller. Moreover, it is envisaged that the performance of the controller could be further improved by implementing any of the suitable alternatives to numerical differentiation for full-state feedback, originally suggested in Section 5.3.4.1, hence avoiding the undesirahle effects of noise amplification inherent in such numerical techniques.

References

[1] CRS Plus Inc. *CRS A100/A200 Series Small Industrial Robot Systems Technical Manual*, (1990).

[2] Unimation (Europe) Ltd. *PUMA Robot Technical Manual*, (1982).

[3] National Intruments Corporation. *Lab-PC+. User Manual*, October (1993).

[4] Hernandez T.A. Industrial manipulator controller interface and system modelling. BEng Honours project, School of Electronic Engineering, Middlesex University, UK, May (1996).

[5] Deneb Robotics Inc. *TELEGRIP Manual*, 3.0 edition, March (1996).

[6] Fu K.S., Gonzalez R.C., and Lee C.S.G. *Robotics: control, sensing, vision, and intelligence.* McGraw-Hill Book Co., (1987).

[7] Kahn M.E. and Roth B. The near-minimum-time control of open-loop articulated kinematic chains. *Journal of Dynamic Systems, Measurement, and Control. Transactions of the ASME*, **93**(3):164–172, September (1971).

[8] Kao C.K., Sinha A., and Mahalanabis A.K. A digital algorithm for near minimum-time control of robot manipulators. *Journal of Dynamic Systems, Measurement, and Control. Transactions of the ASME*, **109**:320–327, December (1987).

[9] Miró J.V., White A.S., and Gill R. On-line time-optimal algorithm for manipulator trajectory planning. In *Proceedings of the European Control Conference*, July (1997). Session TH-E G5, paper number 661 (in CD-ROM).

Chapter 8

Conclusions

The work presented in this thesis has shown the feasibility of applying optimal control for real-time planning and control of robot motions.

The approach is based on the philosophy that incorporating the non-linear manipulator dynamics into the robot planning and control stages can lead to a true maximisation of the manipulator's capabilities at any time instant during the motion. From this generic concept, nurtured during the early stages of the project, the idea of accommodating optimal control theory naturally evolved. A judicious manipulator dynamics approximation by piecewise-linear and decoupled equations of motion in state-space, and the application of Pontryagin's MP, lead to a generic analytical optimal solution to the path unconstrained TPBV manipulator motion problem. By adopting this strategy, the need to resort to numerical, albeit exact, optimal solutions to the problem is avoided, hence rendering a relatively simple controller which brings about near-optimal motions between any two desired end-points. A timing performance has been selected in this work as the measure of optimality in the pursuit of increased manipulator productivity, although the proposed strategy can be easily extended to other performance criteria, e.g. a measure of the energy expended during the defined motion.

8.1 Contributions of this Thesis

The main contributions of this thesis can be summarised as follows:

- A method has been proposed which determines the trajectory and associated control that the manipulator must follow to satisfy given end-point constraints and minimise a time cost function during the motion. Although the application of optimal control theory for the solution of the robot motion problem has been investigated in the literature, very few authors have discussed implementation issues and presented experimental results. Hence, the exploration of alternatives towards an efficient on-line implementation is, indeed, one of the key contributions of this work. To this end, the concept of the averaged dynamics to transform the manipulator dynamics into a piecewise-linear decoupled model in feedback form has been presented. While not a new idea, it has been refined to allow for each boundary condition's dynamic performance to be weighted independently in the estimated overall time-invariant manipulator dynamics. Furthermore, extensive work has been carried out into studying the role of this approximation in the response of the algorithm. It was concluded that a medium-to-small averaged dynamic coefficient (0.4) offered the best performance with a good dynamic range - minimum manipulator dynamics adaptions and adequate overshoot and settling times.

- A Feedforward Controller has been proposed as a novel approach to reduce the chattering oscillations near the steady-state caused by frequent bang-bang control switching. Other strategies have been surveyed in the literature, mainly the use of linear error-driven controllers, but the availability of the steady-state control torques made the proposed strategy a simple alternative. Moreover, it was a natural chattering-removal addition to a direct torque control strategy, such as the one implemented in this work. As a result, remaining torque oscillations in the vicinity of the target state have been quickly damped, hence reducing the overall settling time.

- A desktop virtual reality environment has been employed in the synthesis and simulation of the proposed controller. While not a traditional control analysis package, a computer graphics front-end has greatly accelerated the overall development and testing process, particularly aiding in the understanding of the dynamic system behaviour via simulation software.
- The proposed strategy has been extended to the actual physical system by designing and constructing an in-house digital interface between the master industrial controller, and a stand-alone PC with additional I/O capabilities. This development has allowed the validation of the novel strategy and also the comparison of this strategy with the splined trajectory planner plus PID control implemented in the industrial manipulator. Experimental evidence has shown an average speed-up performances of 14% from maximising the manipulator torques in the near-optimal-time controller. The algorithm was however outperformed by the industrial controller with respect to other evaluation criteria. Hence, steady-state error and damping proved better, if only slightly, under standard control, although this is not detrimental of the proposed motion model, given the "gross" motion specification for which the near-optimal-time controller was synthesised.
- A direct consequence of the model-based characteristic of the controller was the implementation of direct torque control as the driving signal. This is not in itself a novelty, but given the error-driven characteristic of most current industrial manipulators, where the driving voltage signal is proportional to some function of the position error (e.g., PID), an interesting practical contribution about reverse engineering the controller has been made. Although current, and therefore torque, could only but indirectly be controlled through the velocity amplifiers, the validation of the methodology has shown that it is feasible to apply it to the hardware configuration of most current manipulators, without the need to incorporate torque amplifiers, for which the strategy is ideally suited.
- The work presented here can also be regarded, in a more general framework, as a contribution to the practical propositions of what is recently being termed as "soft" real-time control. That is, a new generation of software products that allow the PC to undertake the control functions that would otherwise be assigned to dedicated pieces of embedded systems, such as industrial robot controllers or PLCs. Attraction of an "open" standard development platform, lower hardware cost, access to standard PC software tools, such as high-level computer languages, and large processing power and memory capacities are some of the potential benefits of the technology, which have allowed the development and testing of the proposed strategy.

As a final remark, the author would like to state that the challenge of developing the proposed strategy, like a large percentage of the challenges in advanced control engineering, existed not in the derivation of new theory, but in utilising existing theory to the full in order to achieve the desired objectives.

Summary

The following have been achieved:

1. Clear exposition and rationalisation of current trends in optimal trajectory planning and control.
2. Derivation and assessment of the CRS A251 manipulator dynamic equations of motion.
3. Complete 3D modelling of the robotic system in an advanced graphical simulation environment.
4. Formulation of the near-optimal-time point-to-point control strategy for path unconstrained motions using Pontryagin's MP and modern control system analysis.
5. Simulation of the proposed strategies in the virtual environment.
6. On-line implementation of the near-optimal-time control strategies on a low-cost PC.
7. Design and construction of the controller interface.
8. Validation of simulation results with high accuracy (correlation coefficients $\in [0.86, 0.99]$ depending on the joint and speed of motion)on the PC robot controller.

9. Comparison of experimental and standard industrial controller results, with average speed-ups in the order of 14%.

8.2 Recommendations for Future Work

There are many issues surrounding the topics in this thesis that deserve further study.

1. In this work, a simplistic point-mass dynamic distribution approximation has been considered in the derivation of the manipulator dynamic equations of motion. This was the result of proprietary information provided by the manufacturer about the robot dynamics, which showed to what length the dynamics of the manipulator are overlooked in the arm design, placing more emphasis in the kinematics and electro-mechanical parts. While the validation experiments carried out for the CRS A251 have shown the suitability of the dynamic model, a more thorough study of the inertial distribution mass of the manipulator must be undertaken, and its significance in the overall dynamics studied. In fact, one would ideally like robust identification techniques that can automatically derive such a structure. For instance, in load inertial parameter estimation, one challenge will be to generalize more complex loads than those characterised as point-masses, as assumed in this work. For instance full rigid bodies or time-varying liquids poured into a container. It would then be interesting to compare the results from both models to assess the importance that dynamics detail plays in the efficiency of the algorithm.

2. To some extent related to the manipulator dynamics issue just addressed, it would also be interesting to study the level of complexity in the control structure as a result of the dynamic interactions. The solution implemented in this dissertation has taken the full rigid-body dynamics of the manipulator into account when synthesising the control strategy. This was a deliberate choice, which generalises the solution to a wide range of robot manipulators. However, it is conjectured that if the manipulator exhibits low inertial mechanical construction, and/or high gear ratios, a simpler decoupled dynamic description may suffice to obtain near, or maybe fully optimal performances. While the true dynamics of a robot manipulator could hardly be regarded as constant and linear over the whole workspace of the manipulator, a decoupled approach would, for instance, avoid the need for a torque decoupling approximation, hence significantly simplifying the design and analysis (in particular stability) of the controller.

3. The experiments also pointed out conceptual limitations due to unmodelled nonlinearities in joint 3. Although these are thought to be identified with the light link structure and a preloaded roller chain, these assumptions need to be further explored.

4. The issue of sampling rate also merits re-examination. In the experiments undertaken, this was limited by the 75 MHz PC microprocessor and on-board I/O circuitry to a sampling rate of 250 Hz. While this was stringent enough to carry out all the calculations and minimise the effect of digital sampling in the analogue plant, it also allowed the reading of some spurious data. Furthermore, it is envisaged that a fully blown-up dynamics as described in point 1 will require more processing capabilities, in which case the sample rate would have to be considered with care. It would be interesting to see how the algorithm performs under the new PCs being introduced onto the market, running at frequencies of 300 MHz. On the other hand, thinking further into the future, and given the multi-DoF characteristic of robot manipulators, special purpose real-time Transputers have been acquired in order to test the adequacy of the algorithm for a true parallel and multi-tasking environment.

5. The incorporation of a tachometer to the robot actuators in order to avoid the numerical integration of the positional readings may also be quite useful to derive motor speeds. On the other hand, it has only recently been appreciated that the speed of the servo motor might also be obtained by measuring the period of the CW and anti-CW pulse trains at the controller test points. Hence, the addition of pulse width extraction circuitry to the interface PCBs might suffice for full-state feedback. Otherwise, it would be interesting to consider the design of a state-observer as suggested in Chapter 5.

6. Optimal control has been studied here in the context of arm movements. While its application is not so much an issue in the fine motion of the hand, the adequacy and completeness of the concepts expressed here will certainly have to be tested for optimal hand control.

7. Most of the commercially available industrial robots use harmonic-drive gear trains to transmit the power generated by the motor to the actual robot link. The built-in flexibility that the flexible-spline introduces is a well known fact. This introduces additional degrees of freedom and the dynamics of the manipulator become more complex than predicted by the rigid-body dynamic model, hence possibly incurring dynamic positioning errors, particularly if fast dynamics are excited. The issue of unmodelled resonant frequencies was addressed by analysing the frequency response of the manipulator, where the energy of any residual effect was found to be well beyond the dominant resonant frequencies of the links, and ignored for the purposes of controlling the arm. However, although this issue has been overlooked for gross motion of the robot, accurate modelling of the harmonic-drive (and any other) joint compliance in order to incorporate it into the real-time torque loop control could hopefully lead to a more precise positioning control.

8. The lack of current sensing devices at the time when the PCBs were designed prevented the torque-control loop from being closed. Hence, although the overall control strategy has been closed around the state feedback, it is still open with regards to torque. Experience gained from the implementation of the controller has hinted at some of the advantages that closing the loop would have in the overall controller: DC motor torque ripple effect, amplifier dead-zones around zero torques, input/output non-linear relationships at high torques or parameter variation in electrical components due to temperature are some of the effects that a closed-loop is expected to, at least, attenuate.

9. Another issue that deserves special attention is the domain in which the controller should be designed. As already discussed in Chapter 6, the torque domain is the natural and preferred environment of the control engineer to design direct torque (model-based) controllers. This was also the preferred environment in which to design the near-optimal-time controller. The fact that the actual robot was fitted with voltage amplifiers imposed the need to resort to reverse engineering the controller for direct current control, which proved a successful option on comparison with the standard industrial control strategies. However, practical experiments have shown that direct torque control might actually degrade the optimality of the solution when compared to a voltage-driven controller, something particularly manifested in dominant gravity-induced configurations, where the highest torques are required to move the manipulator. The reasons behind this could probably be attributed to unmodelled motor dynamics in the reverse engineered model of the motor. For instance, several reports in which motor torque constants have been measured showed considerable variation between similar type motors on a single robot, as well as between similar robots. Also, the torque constant reduces with time due to de-magnetisation, and this effect is even hastened when the motor current rating is exceeded. Furthermore, it is a well know fact that during current saturation (in which, incidentally, the optimal controller relies throughout the motion) at high torques, the motor torque constant is not so constant anymore, exhibiting a non-linear behaviour. Another reason which might explain these findings is the slower mechanical time-constant of the motor versus the electrical time response of the system. In view of these facts, it is believed that the domain in which the controller should be designed needs to be further explored and explained.

10. In this work, an initial stability analysis of the plant has been carried out, although simulation (and experimental) results provided the ultimate check on stability. This analytical limitation was partly due to the approximation-related issues of the proposed algorithm, although the limited number of tools available for the theoretical analysis of non-linear systems also posed a major handicap. Theoretical analysis of non-linear systems is, however, a very active area of research, and recent advances in non-linear dynamic state feedback controllers might soon provide a way to find a tractable analysis.

11. The issue of coordinated arm motion, while to some extent defeating the whole purpose of the proposed optimal strategy, can also be explored in the future.

12. The work undertaken in this dissertation has been a successful attempt to close the gap between optimal control theory and application. It can be regarded as a first step towards the design of a truly generic on-line time-optimal trajectory planner/controller. Although this problem should still be considered as an open problem in robotics, mainly hindered by the unavailability of low-cost computational power, the author envisages that, at the current pace of advances in embedded computing resources, this question will also soon be answered. With more and more widespread use of PCs in industrial control, the so-called "soft control", and technological predictions in the order of GHz for processor frequencies in the early 21st century versus current MHz speeds, the increased importance of computing in the robotics and automation community is set to have far-reaching benefits.

Appendix A

Closed Form Dynamics Derivation

The derivation of the customised closed form dynamic equations of the CRS A251 robotic arm are presented here. These expressions are computed for the three gross-motion links of the five DoF revolute manipulator, whose kinematic configuration is sketched in Figure 5.1, and described in detail in Section 5.3.1. The final functions for the closed form dynamics are collected in Equation 5.3.

Following the Lagrangian dynamic formulation of an open-chain mechanical manipulator with rigid links, Equation (5.1), an expression for the kinetic energy of each link i must be first developed. This is given by:

$$K_i = \frac{1}{2} m_i \dot{r}_{m_i}^2 + \frac{1}{2} I_i \dot{\theta}_{m_i}^2 \tag{A.1}$$

where the first term is the kinetic energy due to the linear velocity of the link's center of mass ($\dot{r}_{m_i}$), and the second term is the kinetic energy due to the angular velocity of the link ($\dot{\theta}_{m_i}$) about this center of mass. All quantities are expressed with respect to the robot base frame. The vector r_{m_i} represents the Cartesian location of the center of mass for link i, while θ_{m_i} corresponds to the angular displacement.

Given the assumption of point-mass distribution [1], the inertia tensor at the center of mass for each link, I_i, is the zero matrix. Taking this simplification into consideration, the kinetic energy of each link can be considered in turn by expanding the coordinates and velocity components of the point-mass(es). Hence, expanding the Cartesian coordinates as follows:

$$r_{m_i} = \{x_{m_i}, y_{m_i}, z_{m_i}\} \tag{A.2}$$

the position of the point-mass(es) for each link m_i can be obtained. Thus, referring to Figure 5.1 for kinematic and dynamic components, and employing the notation previously defined in Section 5.3.1, the location of point-mass m_1 can be found to be:

$$\begin{aligned} x_{m_1} &= -lm_1 c_1 \\ y_{m_1} &= -lm_1 s_1 \\ z_{m_1} &= d_1 \end{aligned} \tag{A.3}$$

where d_1 represents the vertical (along the z axis) distance between the origin of the robot frame, and the location of the waist point-mass m_1. By taking derivatives with respect to time, the linear velocity of m_1 is given by:

$$\begin{aligned} \dot{x}_{m_1} &= lm_1 s_1 \dot{\theta}_1 \\ \dot{y}_{m_1} &= -lm_1 c_1 \dot{\theta}_1 \\ \dot{z}_{m_1} &= 0 \end{aligned} \tag{A.4}$$

Squaring (A.4) we obtain:

$$\begin{aligned} \dot{x}_{m_1}^2 &= lm_1^2 s_1^2 \dot{\theta}_1^2 \\ \dot{y}_{m_1}^2 &= lm_1^2 c_1^2 \dot{\theta}_1^2 \\ \dot{z}_{m_1}^2 &= 0 \end{aligned} \tag{A.5}$$

[1] Refer to Section 5.3.1 for a discussion about the implications of this assumption.

Substituting this expression into (A.1), and reducing the solution by trigonometric identities, the total kinetic energy for link 1, K_1, is thus found to be:

$$K_1 = \frac{1}{2} m_1 l m_1^2 \dot{\theta}_1^2 \tag{A.6}$$

It can be seen then how the kinetic energy of a manipulator can be described by a scalar formula as a function of joint position (θ) and velocity ($\dot{\theta}$). The mechanism for deriving the kinetic energy of the rest of the links is the same. However, since the kinematic relationships are obtained with respect to the base frame, the expressions can become noticeably more involved. Hence, for the second link, the location and velocity of the first point mass m_2 is given by:

$$\begin{aligned} x_{m_2} &= l m_2 c_1 c_2 \\ y_{m_2} &= l m_2 s_1 c_2 \\ z_{m_2} &= d_1 - l m_2 s_2 \end{aligned} \tag{A.7}$$

and

$$\begin{aligned} \dot{x}_{m_2} &= -l m_2 (s_1 c_2 \dot{\theta}_1 + s_1 c_2 \dot{\theta}_1) \\ \dot{y}_{m_2} &= l m_2 (c_1 c_2 \dot{\theta}_1 + s_1 s_2 \dot{\theta}_2) \\ \dot{z}_{m_2} &= -l m_2 c_2 \dot{\theta}_2 \end{aligned} \tag{A.8}$$

Raising each coordinate of Equation (A.8) to the power of two, and rewriting the resulting expressions into (A.1), the upper-arm kinetic energy contributed by m_2 is obtained as:

$$K_2(m_2) = \frac{1}{2} m_2 l m_2^2 (c_2^2 \dot{\theta}_1^2 + \dot{\theta}_2^2) \tag{A.9}$$

Because, as shown in Figure 5.1, the upper-arm mass distribution was concentrated in two points, m_2 and m_3, the contribution of m_3 to the total upper-arm kinetic energy is identical to that of m_2, except for the location of m_3 along the upper-arm link. Accordingly,

$$K_2(m_3) = \frac{1}{2} m_3 l m_3^2 (c_2^2 \dot{\theta}_1^2 + \dot{\theta}_2^2) \tag{A.10}$$

The total kinetic energy of joint 2 is made up of the sum of both contributions. That is,

$$K_2 = K_2(m_2) + K_2(m_3) = \frac{1}{2}\dot{\theta}_1^2 (m_2 l m_2^2 + m_3 l m_3^2) c_2^2 + \frac{1}{2}\dot{\theta}_2^2 (m_2 l m_2^2 + m_3 l m_3^2) \tag{A.11}$$

The fact that the fore-arm mass was also distributed in two points, m_4 and m_5, also gives rise to two independent contributions to the total kinetic energy of the link. However, as was the case for the fore-arm, obtaining the kinetic functional structure of one of the point-masses, immediately provides the other one. Hence, the location of m_4 and its first derivative can be expressed as:

$$\begin{aligned} x_{m_4} &= (l m_3 c_2 + l m_4 c_3) c_1 \\ y_{m_4} &= (l m_3 c_2 + l m_4 c_3) s_1 \\ z_{m_4} &= d_1 - l m_3 s_2 + l m_4 s_3 \end{aligned} \tag{A.12}$$

and

$$\begin{aligned} \dot{x}_{m_4} &= -\dot{\theta}_1 s_1 (l m_3 c_2 + l m_4 c_3) - \dot{\theta}_2 c_1 l m_3 s_2 - \dot{\theta}_3 c_1 l m_4 s_3 \\ \dot{y}_{m_4} &= \dot{\theta}_1 c_1 (l m_3 c_2 + l m_4 c_3) - \dot{\theta}_2 s_1 l m_3 s_2 - \dot{\theta}_3 s_1 l m_4 s_3 \\ \dot{z}_{m_4} &= -\dot{\theta}_2 l m_3 c_2 + \dot{\theta}_3 l m_4 c_3 \end{aligned} \tag{A.13}$$

After some cumbersome algebra to reduce the expression, the solution for the kinetic energy of m_4 is given by:

$$\begin{aligned} K_3(m_4) = {} & \frac{1}{2} m_4 \dot{\theta}_1^2 (l m_3^2 c_2^2 + l m_4^2 c_3^2 + 2 l m_3 l m_4 c_2 c_3) + \frac{1}{2} m_4 \dot{\theta}_2^2 l m_3^2 + \\ & \frac{1}{2} m_4 \dot{\theta}_3^2 l m_4^2 - m_4 \dot{\theta}_2 \dot{\theta}_3 l m_3 l m_4 c_{23} \end{aligned} \tag{A.14}$$

And, consequently, the kinetic energy of m_5 is:

$$K_3(m_5) = \frac{1}{2}m_5\dot{\theta}_1^2(lm_3^2c_2^2 + lm_5^2c_3^2 + 2lm_3lm_5c_2c_3) + \frac{1}{2}m_5\dot{\theta}_2^2lm_3^2 + \frac{1}{2}m_5\dot{\theta}_3^2lm_5^2 - m_5\dot{\theta}_2\dot{\theta}_3lm_3lm_5c_{23} \quad (A.15)$$

Hence, the total kinetic energy for the fore-arm can be combined as follows:

$$K_3 = K_3(m_4) + K_3(m_5) = \frac{1}{2}\dot{\theta}_1^2\{(m_4+m_5)lm_3^2c_2^2 + (m_4lm_4^2 + m_5lm_5^2)c_3^2 + 2lm_3(m_4lm_4 + m_5lm_5)c_2c_3\} + \frac{1}{2}\dot{\theta}_2^2(m_4+m_5)lm_3^2 + \frac{1}{2}\dot{\theta}_3^2(m_4lm_4^2 + m_5lm_5^2) - \dot{\theta}_2\dot{\theta}_3lm_3(m_4lm_4 + m_5lm_5)c_{23} \quad (A.16)$$

The total kinetic energy of the manipulator is the sum of the kinetic energy in the individual links. For the three DoFs being analysed, that is,

$$K = \sum_{i=1}^{3} K_i \quad (A.17)$$

Having developed an expression for the manipulator kinetic energy, the potential energy of the robot arm must then be obtained in order to derive the Lagrangian of the manipulator (see Equation 5.1). Potential forces are represented by gravity forces, thus the potential energy of each link can be expressed as:

$$P_i = m_i\boldsymbol{g}\boldsymbol{h}_{m_i} \quad (A.18)$$

where $\boldsymbol{g}$ is the gravity vector and $\boldsymbol{h}_{m_i}$ is the Cartesian vector locating the center of mass of the ith link. As before, for a point-mass distribution, the potential energy can be obtained by applying (A.18) to each point-mass m_i. Furthemore, if the robot is upright, the gravitational acceleration vector is proportional to the z component of the location vector only. Hence, for m_1, the potential energy can be reduced to the following scalar quantity, where $\boldsymbol{h}_{m_1}$ is proportional to z_{m_1} in Equation (A.3), as follows:

$$P_1 = m_1gd_1 \quad (A.19)$$

Combining the vertical Cartesian location of m_2, as given by (A.7), with Equation (A.18), the potential energy of the upper-arm stored in m_2 can be derived as:

$$P_2(m_2) = m_2g(d_1 - lm_2s_2) \quad (A.20)$$

Notice that bacause $\boldsymbol{h}_{m_i}$ in (A.18) is described as a function of θ, the potential energy of a manipulator can be described by a scalar formula as a function of the joint positions. Accordingly, the potential energy in m_3 is given by:

$$P_2(m_3) = m_3g(d_1 - lm_3s_2) \quad (A.21)$$

So that the total potential energy of link 2 can be obtained as:

$$P_2 = P_2(m_2) + P_2(m_3) = d_1g(m_2+m_3) - s_2g(lm_2m_2 + lm_3m_3) \quad (A.22)$$

Analagously, by substituting the z component of m_4 in Equation (A.12) - and the equivalent of m_5 - in (A.18), the potential energy stored in these point-masses can be derived as:

$$P_3(m_4) = m_4g(d_1 - lm_3s_2 + lm_4s_3)$$
$$P_3(m_5) = m_5g(d_1 - lm_3s_2 + lm_5s_3) \quad (A.23)$$

Combining (A.23) into one, the potential energy of the fore-arm can be then expressed as:

$$P_3 = P_3(m_4) + P_3(m_5) = d_1g(m_4+m_5) - s_2lm_3g(m_4+m_5) + s_3g(lm_4m_4 + lm_5m_5) \quad (A.24)$$

The total potential energy stored in the manipulator can be now obtained as the sum of the potential energy in the individual links; that is,

$$P = \sum_{i=1}^{3} P_i \quad (A.25)$$

The derivation of the manipulator equations of motion can be now addressed in two alternative ways:

1. Jointly for the overall Lagrange function of the manipulator, L, as described by Equation (5.1). That is, L is first obtained as the subtraction of (A.25) from (A.17). Then, partial derivatives of this expression with respect to each generalised coordinate, θ_1, θ_2 and θ_3, and their first time derivatives are taken to obtain τ_1, τ_2 and τ_3 respectively.

2. Separately for the kinetic and gravity forces of each link.

While the first option is faster to derive, it also loses part of the insight into the contribution of the different forces to the overall torque developed by each motor. Furthermore, given the complexity of the resulting expression for L, the subsequent derivation procedure is more inclined to errors. Hence, the second mechanism was preferred in deriving the dynamic equations of the CRS A251.

In the development presented next, the notation τ_{ij} implies the torque that has to be exerted by the motor at joint i to support the motion and gravity-induced forces of link j. Due to the mechanical construction of the manipulator, some important preliminary conclusions about the structure of the Lagrange-Euler equation of motion, given by formula (5.1), can be drawn prior to its derivation for the particular case of the CRS A251:

- The potential energy is a function of θ_i, but not $\dot{\theta}_i$. Hence, $\partial P_i/\partial \dot{\varrho}_i = 0$ for all i.
- There is no contribution of a motor link to support motion of lower links, i.e., $\tau_{ij} = 0$ if $i > j$.

Taking these issues into consideration, the torques developed by motors at joint 1, 2 and 3 will now be analysed in turn.

Waist motor (joint 1) The torque components to be developed by the waist joint motor to support motion and gravity-induced forces of its own link (τ_{11}), upper-arm (τ_{12}) and fore-arm (τ_{13}) links are, respectively (and after some trigonometric identities):

$$\tau_{11} = \frac{d}{dt}\left(\frac{\partial L_1}{\partial \dot{\varrho}_1}\right) - \frac{\partial L_1}{\partial \varrho_1} = \frac{d}{dt}\left(\frac{\partial K_1}{\partial \dot{\varrho}_1}\right) = \ddot{\theta}_1\{m_1 l m_1^2\} \tag{A.26}$$

$$\tau_{12} = \frac{d}{dt}\left(\frac{\partial L_2}{\partial \dot{\varrho}_1}\right) - \frac{\partial L_2}{\partial \varrho_1} = \frac{d}{dt}\left(\frac{\partial K_2}{\partial \dot{\varrho}_1}\right) = \ddot{\theta}_1\{c_2^2(m_2 l m_2^2 + m_3 l m_3^2)\} - \dot{\theta}_1\dot{\theta}_2\{2 s_2 c_2 (m_2 l m_2^2 + m_3 l m_3^2)\} \tag{A.27}$$

$$\begin{aligned}\tau_{13} = \frac{d}{dt}\left(\frac{\partial L_3}{\partial \dot{\varrho}_1}\right) - \frac{\partial L_3}{\partial \varrho_1} = \frac{d}{dt}\left(\frac{\partial K_3}{\partial \dot{\varrho}_1}\right) &= \ddot{\theta}_1\{(m_4 + m_5) l m_3^2 c_2^2 + (m_4 l m_4^2 + m_5 l m_5^2) c_3^2 + \\ &(m_4 l m_4 + m_5 l m_5) 2 l m_3 c_2 c_3\} - \\ &\dot{\theta}_1\dot{\theta}_2\{(m_4 + m_5) 2 l m_3^2 s_2 c_2 + (m_4 l m_4 + m_5 l m_5) 2 l m_3 s_2 c_3\} - \\ &\dot{\theta}_1\dot{\theta}_3\{(m_4 l m_4^2 + m_5 l m_5^2) 2 s_3 c_3 + (m_4 l m_4 + m_5 l m_5) 2 l m_3 c_2 s_3\}\end{aligned} \tag{A.28}$$

Expressions (A.26), (A.27) and (A.28) can be now added together to obtain an expression for the overall torque to be developed by the actuator in joint 1, i.e.,

$$\tau_1 = \tau_{11} + \tau_{12} + \tau_{13} \tag{A.29}$$

which, after some re-arranging, gives the customised expression for τ_1 in Equation (5.3).

Upper-arm motor (joint 2) As before, the torques required from the motor of joint 2 to support the forces exerted at link 2 and 3 can be obtained as:

$$\begin{aligned}\tau_{22} = \frac{d}{dt}\left(\frac{\partial L_2}{\partial \dot{\varrho}_2}\right) - \frac{\partial L_2}{\partial \varrho_2} = \frac{d}{dt}\left(\frac{\partial K_2}{\partial \dot{\varrho}_2}\right) - \left(\frac{\partial K_2}{\partial \varrho_2} - \frac{\partial P_2}{\partial \varrho_2}\right) &= \ddot{\theta}_2\{m_2 l m_2^2 + m_3 l m_3^2\} + \\ &\dot{\theta}_1^2\{(m_2 l m_2^2 + m_3 l m_3^2) s_2 c_2\} - g\{(m_2 l m_2 + m_3 l m_3) c_2\}\end{aligned} \tag{A.30}$$

$$\begin{aligned}\tau_{23} = \frac{d}{dt}\left(\frac{\partial L_3}{\partial \dot{\varrho}_2}\right) - \frac{\partial L_3}{\partial \varrho_2} = \frac{d}{dt}\left(\frac{\partial K_3}{\partial \dot{\varrho}_2}\right) - \left(\frac{\partial K_3}{\partial \varrho_2} - \frac{\partial P_3}{\partial \varrho_2}\right) &= \ddot{\theta}_2\{(m_4 + m_5) l m_3^2\} - \\ &\ddot{\theta}_3\{(m_4 l m_4 + m_5 l m_5) l m_3 c_{23}\} + \dot{\theta}_1^2\{(m_4 + m_5) l m_3^2 s_2 c_2 + (m_4 l m_4 + m_5 l m_5) l m_3 s_2 c_3\} + \\ &\dot{\theta}_3^2\{(m_4 l m_4 + m_5 l m_5) l m_3 s_{23}\} - g\{(m_4 + m_5) l m_3 c_2\}\end{aligned} \tag{A.31}$$

These torque components are to be added to derive the overall torque to be developed by the actuator in joint 2, τ_2, whose final expression is given in Equation (5.3). That is,

$$\tau_2 = \tau_{22} + \tau_{23} \tag{A.32}$$

Fore-arm motor (joint 3) The motor at joint 3 needs only to develop a torque to counteract motion and gravity-induced forces exerted by its own link. The dynamic expression for this torque can be obtained as:

$$\begin{aligned}\tau_3 = \tau_{33} = & \frac{d}{dt}\left(\frac{\partial L_3}{\partial \dot{\varphi}_3}\right) - \frac{\partial L_3}{\partial \varphi_3} = \frac{d}{dt}\left(\frac{\partial K_3}{\partial \dot{\varphi}_3}\right) - \left(\frac{\partial K_3}{\partial \varphi_3} - \frac{\partial P_3}{\partial \varphi_3}\right) = \\ & -\ddot{\theta}_2\{(m_4 l m_4 + m_5 l m_5) l m_3 c_{23}\} + \\ & \ddot{\theta}_3\{(m_4 l m_4^2 + m_5 l m_5^2)\} + \dot{\theta}_1^2\{(m_4 l m_4^2 + m_5 l m_5^2) s_3 c_3 + (m_4 l m_4 + m_5 l m_5) l m_3 s_3 c_2\} + \\ & \dot{\theta}_2^2\{(m_4 l m_4 + m_5 l m_5) l m_3 s_{23}\} + g\{(m_4 l m_4 + m_5 l m_5) c_3\}\end{aligned} \tag{A.33}$$

This torque solution can be further manipulated to obtain the reduced expression of τ_3 in Equation (5.3).

Appendix B

Source Code

This Appendix presents the source code developed in ANSI C with Borland Turbo C++ to control the manipulator from an on-line PC. It provides simultaneous control of the three major links of the robot via an equal number of DAQ cards. Refer to Sections 7.2 and 7.4 for more details about the actual HW/SW implementation. The code is structured into the following functional blocks:

- `Crs_params.h` Declaration of the CRS A251 kinematic, dynamic, and actuating subsystem parameters.
- `Crs_defs.h` Declaration of global data structures, DAQ definitions, function prototypes, auxiliary constants and global C header files.
- `Crs_main.c` Main routine performing, in a continuous loop, state input + computation of control action + control output.
- `Crs_init.c` Structure and DAQ initialisation routines. Dynamic memory allocation.
- `Dynamics.c` Computation of the CRS A251 dynamic equations of motion and auxiliary procedures.
- `Control.c` Routines to compute the near-optimal-time torque curves for current state and classical PID control action.
- `Daq_ctrl.c` Procedures to send control action through the data acquisition cards.

```
/*****************************************************************************
 *                    Module Crs_params.h                                    *
 *                                                                           *
 *                     Jaime Valls Miro                                      *
 *****************************************************************************/
#define   g             9.81    /* Metric(S.I.)system -> [m/(s^2)] */
       /* Link characteristics */
#define DoF             3       /* Gross motion links only*/
#ifdef CRS_AVG_MASSES
       /* Metric(S.I) system --> Link point Masses in kg. */
       #define   m1  4.35       /* Waist     = m1      */
       #define   m2  0.00       /* Upper arm = m2 + m3 */
       #define   m3  1.73
       #define   m4  0.00       /* Fore arm  = m4 + m5 + m6 + m7(load)*/
       /* Wrist + gripper + load mass considered located at end of Fore-arm -> mS
        * mS  0.36 ;  m6  0.23 ; m7  1.00 --> Max. nominal load of 1kg(2.20lb) */
       #define . m5   2.21     /* 0.62+0.36+0.23+1.0 */
       /* Metric(S.I) system --> Link Lenghts in m. */
       #define  lm1  0.135     /* Waist     = lm1 */
       #define  lm2  0.0
       #define  lm3  0.18415   /* Upper arm = lm3 = 7.25 in*/
```

```
      #define  lm4  0.0
      #define  lm5  0.1905    /* Fore arm  = lm5 = 7.5 in*/
#endif
#ifdef CRS_DIST_MASSES
       /* Metric(S.I) system --> Link point Masses in kg. */
       #define   m1  4.35      /* Waist     = m1      */
       #define   m2  0.84      /* Upper arm = m2 + m3 */
       #define   m3  0.89
       #define   m4  0.62      /* Fore arm  = m4 + m5 + m6 + m7(load)*/
       /* Wrist + gripper + load mass considered located at end of fore-arm -->m5
        * m5  0.36 ; m6  0.23 ; m7(load)  0.0 --> Max. nominal load of 1kg(2.20lb) */
      #define   m5_noload     0.59
      #define   load          0.0
      #define   m5  (m5_noload+load)
      /* Metric(S.I) system --> Link Leaghts in m. */
      #define  lm1  0.135      /* Waist     = lm1 */
      #define  lm2  0.114
      #define  lm3  0.254      /* Upper arm = lm3 */
      #define  lm4  0.127
      #define  lm5  0.254      /* Fore arm  = lm5 */
#endif
       /* Actuator characteristics */
#define         N               72              /* Gear ratio */
static  float   gear_eff[DoF]={0.75,0.75,0.56};/* Gear efficiencies */
static  float   Rm[DoF] = {4.4,4.4,4.4};        /* Armature resistance (Ohms)[4.0, 6.0]*/
#define         Jm              0.000028        /* Jm = 0.085 lbf*in^2 (* 1/32 s^2/ft *
                                                 * 1/12^2ft^2/in^2*1.35kg*m^2/lbf*s^2*ft)
                                                 * = 0.000025 kg*m^2 + 0.0116 lbf*in^2
                                                 * for HD wave generator */
static float    Jmeff[DoF][DoF]={{0.109,0    ,0    },  /* Jmeff = Jm*N^2*gear_eff[i] */
                                 {0    ,0.109,0    },
                                 {0    ,0    ,0.081}};
#define    L                 0.004  /* Motor inductance */
#define    Km                0.0657 /* Motor torque constant Nm/Amp */
#define    Kbemf             0.0657 /* Back e.m.f. constant  V*s/rad */
#define    motor_torque_max  0.128  /* +/-0.085 not to exceed motor MAX.VEL.RATING
                                     * Maximum CRS actuators output torque [N*m]
                                     * motor torque max=0.128N*m=18Oz*in^9.3Oz*in/Amp
                                     * x 2 Amp (max motor current) */
#define    max_PC_voltage    4.99   /* Maximum voltage provided by LabPC+ */
#define    max_motor_voltage 19.99  /* Maximum voltage from LA to motor */
#define    FINAL_TOLER_THETAS 0.0087 /* [rad] error w.r.t. final pos->End motion */
#define    FINAL_TOLER_SPEEDS 0.02   /* [rad/s] error w.r.t. final vel.->"     */
#define    OPT_TOLER_THETAS   0.027  /* [rad]=0.5deg error w.r.t. final pos->End nearopt
                                     * dyn-model-based ctrl+start feedforward ctrl   */
#define    OPT_TOLER_SPEEDS   0.27   /* [rad/s]=10deg/s error w.r.t. final vel.->"    */

/*****************************************************************************
 *                      Module crs_defs.h                                    *
 *                                                                           *
 *                      Jaime Valls Miro                                     *
 *****************************************************************************/
#include <nidaq.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
```

```
#include <time.h>
#include <conio.h>
        /* Conditional Compilation definitions */
#define CRS_DIST_MASSES
#define FeedForward     /* Alternatives: PID, ComputerTorque, FeedForward_NoModel */
#include <crs_parm.h>
        /* DAQ constants */
        /* LabPC+ devices: joint1=1, joint2=2, joint3=3 */
        /* Analog Ports */
#define DAC0            0       /* Analog Output */
        /* Digital Ports */
#define DPortA          0       /* Data Input */
#define DPortB          1       /* Control Outout */
#define DPortC          2       /* Control Output */
        /* Digital Lines */
#define Line0           0       /* Count Clock Up or Down */
#define Line1           1       /* Clear everything */
#define Line2           2       /* Load initial value */
#define Line3           3       /* Count Down or Up constantly at high level */
#define read            0
#define write           1
        /* General constants */
#define FALSE                   0
#define TRUE                    !FALSE
#define mmTOm                   0.001
#define PI                      3.1416
#define rad_to_deg              (180/PI)
#define deg_to_rad              (PI/180)
#define pulsestoradians         (0.005*deg_to_rad)/* 0.005 link_deg/mot_pulse */
#define samples                 700             /* state samples in out file */
        /* DOS timing */
#define TMR_CNT 20      /*  New frequency from 1.192737 MHz/TMR_CNT */
                        /*  The original being 1.192737 MHz/65535=18.2 Hz */
                        /*  1193 = 1000Hz (1 kHz) */
                        /*   239 = 5000Hz (5 kHz) */
                        /*   119 = 10000Hz(10 kHz) */
                        /*    24 = 50000Hz(50 kHz) */
                        /*    20 = 60000Hz(60 kHz) --> Maximum !  */
#define ticks_sample            239.0
#define sample_time             (ticks_sample*TMR_CNT/1.192737E+6)
                        /* = 0.004 s of sampling time -> Minimum ! */
/* Global type definitions */
typedef struct
        {       float   max_req_time;
                float   *alpha_final;
                float   *beta_final;
                float   *alpha_avg;
                float   *beta_avg;
                float   *torque_last;
                float   *holding_torque;
                float   *prev_thetas;
                float   *prev_prev_thetas;
                float   *prev_prev_prev_thetas;
        } mot_data_struct;
typedef struct
        {       float err_qq [DoF];       /* current control error */
```

```
                float  err_ii [DoF];
                float  err_vv [DoF];
                float  prev_err_qq [DoF];   /* previous control error */
                float  prev_err_ii [DoF];
        } pid_ctrl_data_struct;
        /* Global headers for auxiliary functions */
        /* Functions declared in file crs_init.c */
void crs_init( mot_data_struct *mot_data, pid_ctrl_data_struct *pid_data );
void crs_init_daq( void );
short configure_ports(short device);
short counter_init_state(short device);
short clear_and_load_init_value(short device);
        /* Functions declared in file dynamics.c */
void inertia_matrix( float *thetasp, float (*D)[DoF]);
void inverse_inertia_matrix(float (*D)[DoF], float (*D_inv)[DoF]);
void centripetal_coriolis_and_gravity_matrix(float *thetasp,float *speedsp,float *HG);
void calc_alpha( float *alpha, float (*D_inv)[DoF] );
void calc_beta( float *beta, float (*D_inv)[DoF], float *HG, float *torques );
void calc_torque(float *torques, float *accels, float (*D)[DoF], float *HG );
        /* Functions declared in file control.c */
void torque_curve(int joint, mot_data_struct *mot_data, float *final_thetas,
                float *speedsp, float *thetasp);
float pid_ctrl( pid_ctrl_data_struct *pid_data,int joint, float *thetasp,
                float *final_thetas);
        /* Functions declared in file daqctrl.c */
float send_torque_control_voltage( float *torque, float *speedsp,
                mot_data_struct *mot_data, short joint);
void send_PID_control_voltage(float torque_error, float *current_speeds,short joint);
        /* Macro definitions  */
#define check_torque_limits(curr_error_torque, max_torque)
        (fabs(curr_error_torque) > max_torque ? ( curr_error_torque > 0.0 ?
        max_torque : -max_torque ) : curr_error_torque )
#define sign(vel)( vel > 0.001 ? 1.0 : -1.0 ) /* Sign of velocity */
#define moving(vel) ( fabs(vel) > 0.01 ? 1.0 : 0.0 ) /* Is joint moving? */

/*****************************************************************************
 *                      Module crs_main.c                                    *
 *                                                                           *
 *                      Jaime Valls Miro                                     *
 *****************************************************************************/
#include <crs_defs.h>
        /* Global variables definitions */
float   *initial_thetas,*final_thetas,*initial_speeds,*final_speeds,
        *initial_accels,*final_accels, *alpha,*beta, offset_theta[DoF],
        *torque_array, avg_factor;
int     final_pos_reached[DoF] = { TRUE, TRUE, TRUE },
        opt_pos_reached[DoF] = { TRUE, TRUE, TRUE };
short   count[DoF], countA, countB;      /*   current pos. pulses */
float   torque_max[DoF]; /* = {6.9,6.9,5.16} Maximum "mathematical"(constant) CRS
                          * actuator output torque [Nm]
                          * 0.128 N*m (motor_torque_max)x72(gear ratio)xgear_eff[i]*/
double                  Vm[samples][DoF];
mot_data_struct         *mot_data;
pid_ctrl_data_struct    *pid_data;
FILE                    *state_fp;
        /* Here due to stack overflow, but no need for global variables */
```

```
float   current_thetas[samples][DoF],     /* [rad] */
        current_speeds[samples][DoF],
        current_accels[samples][DoF],
        current_torques[samples][DoF],
        measure_torque[samples],temp_thetas[DoF],tot_time;
int     i=1;
        /* Change timer */
void higher_res()
{
asm mov al, 0x36;
asm out 0x43, al;
asm mov ax, TMR_CNT;
asm out 0x40, al;
asm mov al, ah;
asm out 0x40,al;
}
void normal_res()
{
int tmr=0;
asm mov al, 0x36;
asm out 0x43, al;
asm mov ax, tmr;
asm out 0x40, al;
asm mov al, ah;
asm out 0x40,al;
}

/*-------------------------------------------------------------------------
 * Main controller program starts here-------
 *-------------------------------------------------------------------------*/
void main()
{
int     joint, j;
float   accum_time[samples], D[DoF][DoF], D_inv[DoF][DoF],HG[DoF],temp,
        current_error_torque, beta_file[samples][DoF], alpha_file[samples][DoF];
char    c,cnl,keyb;
short   init_joint = 0, daqErr;
double  voltin;
clock_t time_1, time_2, ticks_for_next_sample, start_time;

printf("crs_main() entered");
        /* Initialize various param's and DAQ PC cards at beginning of motion. */
mot_data = (mot_data_struct *) malloc(sizeof(mot_data_struct));
pid_data = (pid_ctrl_data_struct *) malloc(sizeof(pid_ctrl_data_struct));
crs_init(mot_data, pid_data);
        /* Calculate current maximum torques */
for(joint = init_joint;joint < DoF;joint++)
        torque_max[joint] = (motor_torque_max*N*gear_eff[joint]);
        /* Start computation cycle for each joint until final position is reached.
         * When joint reaches final position only feedforward control is applied */
for( joint = init_joint; joint < DoF ; joint++ )
{
        current_thetas[0][joint] =  initial_thetas[joint];
        current_speeds[0][joint] =  initial_speeds[joint];
        current_accels[0][joint] = initial_accels[joint];
        current_torques[0][joint] =  mot_data->torque_last[joint];
```

```
        Vm[0][joint] =  mot_data->torque_last[joint];
}
        /* Hold robot at initial configuration */
for(joint = init_joint;joint < DoF;joint++)
        Vm[0][joint ] = send_torque_control_voltage( current_torques+0,
                        current_speeds+0, mot_data ,joint);
        /* Initial synchronisation */
printf("Resetting system. Hit Enter to continue\n");
scanf("%c",&c);
        /* Finer DOS clock resolution */
higher_res();
accum_time[0] = measure_torque[0] = 0.0;
do{
      ticks_for_next_sample = clock()+ticks_sample;
      do{
        /* Synchronisation loop */
        while(clock() <= ticks_for_next_sample)
                /* do nothing */;
        accum_time[i] = accum_time[i-1]+sample_time;
        tot_time = accum_time[i];
        time_1 = clock();
        /* Current joint axes values at begianing of current iteration */
        for(joint=init_joint;joint<DoF;joint++) /* i=1=joint2, i=2=joint_3 */
        {
            daqErr=DIG_In_Port(joint+1, DPortA, &countA);
            daqErr=DIG_In_Port(joint+1, DPortB, &countB);
            count[joint]=(countB*256+countA);
            count[joint]=(count[joint]> 32766)? -65535+count[joint]:count[joint];
            current_thetas[i][joint] =(count[joint]*pulsestoradians) +
                offset_theta[joint];
            /* Numerical 2 point velocity approximation. */
            current_speeds[i][joint] = (current_thetas[i][joint] -
                                        mot_data->prev_thetas[joint])/sample_time;
            current_speeds[i][joint]*= 0.6;
            current_speeds[i][joint]+= current_speeds[i-1][joint] * 0.4;
            /* Numerical 2 point acceleration approximation. */
            current_accels[i][joint] = (current_speeds[i][joint] -
                                        current_speeds[i-1][joint])/sample_time;
            current_accels[i][joint]*= 0.15;
            current_accels[i][joint]+= current_accels[i-1][joint] * 0.85;
            mot_data->prev_thetas[joint]=current_thetas[i][joint];
            Vm[i][joint]=0.0;
        }
        /* Safety stop if Waist  Joint (0) gone beyond limits */
        if(fabs(current_thetas[i][0]) > 1.7)
        {
            for(joint = 1; joint <= DoF ; joint++)
                    daqErr = AO_VWrite (joint, DAC0, 0.0);
            break;
        }
        /* Update each of the joint axes dynamic coefficients.
         * Update alpha's */
        inertia_matrix(current_thetas+i, D);
        inverse_inertia_matrix( D, D_inv );
        calc_alpha( alpha, D_inv );
        /* Update beta's */
```

```
centripetal_coriolis_and_gravity_matrix(current_thetas+i,current_speeds+i,HG);
calc_beta( beta, D_inv, HG, mot_data->torque_last );
/* Calculate average dyn. coefficients for linear eq. of motion */
for( joint = init_joint; joint < DoF ; joint++ )
{
    mot_data->alpha_avg[joint] = (1-avg_factor)*alpha[joint] +
                                 avg_factor*mot_data->alpha_final[joint];
    alpha_file[i][joint] = mot_data->alpha_avg[joint] ;
    mot_data->beta_avg[joint] = (1-avg_factor)*beta[joint] +
                                avg_factor*mot_data->beta_final[joint];
    beta_file[i][joint] = mot_data->beta_avg[joint] ;
}
for( joint = init_joint; joint < DoF ; joint++ )
{
    if(opt_pos_reached[joint])
    {
        if((fabs(final_thetas[joint]-current_thetas[i][joint])<=FINAL_TOLER_THETAS)
        &&(fabs(final_speeds[joint]-current_speeds[i][joint])<=FINAL_TOLER_SPEEDS))
                final_pos_reached[joint]=TRUE;
    /* Calculate Linear non-model-based PID control action
     * from "measured state"(ON-LINE) and send it down the DA cards */
     #ifdef PID
        Vm[i][joint] = current_error_torque = pid_ctrl(pid_data, joint,
                                                current_thetas+i,final_thetas );
        send_PID_control_voltage(current_error_torque, joint);
     #endif
    /* Calculate Computed Torque control action
     * from "measured state"(ON-LINE) and send it down the DA cards */
     #ifdef ComputerTorque
        current_accels[joint]=pid_ctrl(pid_data,joint,current_thetas+i,
                final_thetas);
        current_error_torque = 0.0;
        for(j = 0; j < DoF ; j++)
                current_error_torque+= D[joint][j]*current_accels[j];
        current_torques[i][joint] = current_error_torque+=HG[joint];
        Vm[i][joint] = send_torque_control_voltage( current_torques+i,
                current_speeds+i, mot_data ,joint);
     #endif
    /* Calculate feedforward control action from "desired positions"
     * (OFF-LINE = holding torque of final position & vel.) +
     * send down DA cards WITHOUT actuator model */
     #ifdef FeedForward_NoModel
        Vm[i][joint] = current_error_torque = mot_data->holding_torque[joint]+
                pid_ctrl(pid_data, joint,current_thetas+i,final_thetas);
        send_PID_control_voltage(current_error_torque, current_speeds+i, joint);
     #endif
    /*
     * Calculate feedforward control action
     * from "desired positions"(OFF-LINE=holding torque at final pos & vel)
     * and send it down the DA WITH actuator model
     */
     #ifdef FeedForward
        current_torques[i][joint] = mot_data->holding_torque[joint] +
                pid_ctrl(pid_data, joint, current_thetas+i, final_thetas ) ;
        mot_data->torque_last[joint]=Vm[i][joint]=send_torque_control_voltage(
                current_torquee+i, current_speeds+i, mot_data ,joint);
```

```
 #endif
}
else
{
    if((fabs(final_thetas[joint]-current_thetas[i][joint])<=OPT_TOLER_THETAS)
    &&(fabs(final_speeds[joint]-current_speeds[i][joint])<=OPT_TOLER_SPEEDS))
    {
        opt_pos_reached[joint]=TRUE;
    /* Calculate new control action from chattering damping alg.*/
    /* Calculate Linear non-model-based PID control action
     * from "measured state"(ON-LINE) and send it down the DA cards */
     #ifdef PID
        Vm[i][joint] = current_error_torque = pid_ctrl(pid_data, joint,
            current_thetas+i, final_thetas ) ;
        send_PID_control_voltage(current_error_torque, joint);
     #endif
    /* Calculate Computed Torque control actionfrom "measured state"
     * (ON-LINE) and send it down the DA cards */
     #ifdef ComputerTorque
        current_accels[joint] = pid_ctrl(pid_data, joint,
                    current_thetas+i, final_thetas ) ;
        current_error_torque = 0.0;
        for(j = 0; j < DoF ; j++)
                current_error_torque+= D[joint][j]*current_accels[j];
        current_torques[i][joint] = current_error_torque+= HG[joint];
        Vm[i][joint] = send_torque_control_voltage(current_torques+i,
                    current_speeds+i, mot_data ,joint);
     #endif
    /* Calculate feedforward control action
     * from "desired positions" (OFF-LINE=holding torque of final pos & vel)
       + send it down the DA WITHOUT actuator model */
     #ifdef FeedForward_NoModel
        Vm[i][joint]=current_error_torque=mot_data->holding_torque[joint]+
                     pid_ctrl(pid_data,joint,current_thetas+i,final_thetas);
        send_PID_control_voltage(current_error_torque,current_speeds+i,joint);
     #endif
    /* Calculate feedforward control action
     * from "desired positions" (OFF-LINE=holding torque of final pos & vel.)
     * + send it down the DA WITH actuator model */
     #ifdef FeedForward
        current_torques[i][joint] = mot_data->holding_torque[joint] +
                    pid_ctrl(pid_data,joint,current_thetas+i, final_thetas);
        Vm[i][joint]=send_torque_control_voltage(current_torques+i,
                    current_speeds+i,mot_data,joint);
     #endif
    }
    else
    {
    /* Calculate near-optimal control curves and current action */
        torque_curve(joint,mot_data,final_thetas,current_speeds+i,
         current_thetas+i);
        current_torques[i][joint] = mot_data->torque_last[joint];
        mot_data->torque_last[joint]=Vm[i][joint]=send_torque_control_voltage(
         current_torques+i, current_speeds+i, mot_data, joint);
    }
}
```

```
        }
       i++;
       ticks_for_next_sample = time_1+ticks_sample;
       }while( !kbhit() );
       /* Another motion? */
printf("Hit 'c' to continue, 'q' to quit: ");
scanf("%c", &cnl);      /* Discard newline char */
scanf("%c", &c);
}while(c!='q');
       /* Resetting output on exit */
for(joint = init_joint;joint < DoF;joint++)
{
       daqErr = AO_VWrite (joint+1, DAC0, 0.0 );
       if (daqErr)
            printf(" Error #:%d Reseting device: %i\n", daqErr, joint);
}
       /* Update+close state output file */
for(j = 0;j < i-1;j++)
{
  fprintf(state_fp,"%6.4f  ",accum_time[j]);
  for(joint = init_joint; joint < DoF; joint++)
  {
       fprintf(state_fp," %6.4f   %6.4f   %6.4f  ",current_thetas[j][joint],
            current_speeds[j][joint], current_accels[j][joint] );
  }
  fprintf(state_fp," %6.4f  \n ",measure_torque[j]);
}
fclose(state_fp);
       /* Free memory allocated dynamically */
free(alpha);
alpha = NULL;
free(beta);
beta = NULL;
       /* Set PC counter back to normal resolution */
normal_res();
       /* exit */
printf("crs_main() exited\n");
return;
}      /* end function */

/*****************************************************************************
 *                    Module crs_init.c                                      *
 *                                                                           *
 *                    Jaime Valls Miro                                       *
 *****************************************************************************/
#include<crs_defs.h>
       /* External declarations */
extern float    *initial_thetas,*final_thetas,*initial_speeds,*final_speeds,
                *initial_accels,*final_accels,*alpha,*beta,*torque_array,
                avg_factor,offset_theta[DoF];
extern short    countA, countB;
extern FILE     *state_fp;

/*---------------------------------------------------------------------------
 * Procedure to initialise data structures and all I/O boards
 *---------------------------------------------------------------------------*/
```

```
void crs_init(mot_data_struct *mot_data,pid_ctrl_data_struct *pid_data)
{
int     i, j, joint;
char    nl;
float   D[DoF][DoF],D_inv[DoF][DoF], HG[DoF];

printf("crs_init() entered\n");
        /* Initialize PC DAQ boards (LabPC+ 1,2 & 3) */
crs_init_daq();
        /* Initialize file for state output */
if( (state_fp = fopen("C:\\jvm_test\\state.txt","w")) == NULL)
        printf("Error openning output state file\n");
        /* Allocate memory for mot_data array pointers */
mot_data->alpha_final          = (float *) calloc( DoF, sizeof( float ) );
mot_data->beta_final           = (float *) calloc( DoF, sizeof( float ) );
mot_data->alpha_avg            = (float *) calloc( DoF, sizeof( float ) );
mot_data->beta_avg             = (float *) calloc( DoF, sizeof( float ) );
mot_data->torque_last          = (float *) calloc( DoF, sizeof( float ) );
mot_data->holding_torque       = (float *) calloc( DoF, sizeof( float ) );
mot_data->prev_thetas          = (float *) calloc( DoF, sizeof( float ) );
mot_data->prev_prev_thetas     = (float *) calloc( DoF, sizeof( float ) );
mot_data->prev_prev_prev_thetas = (float *) calloc( DoF, sizeof( float ) );
        /* Allocate memory to dyn. coef's array pointers */
alpha = (float *)calloc(DoF, sizeof(float));
beta = (float *)calloc(DoF, sizeof(float));
        /* Allocate memory to motion data arrays */
initial_thetas  = (float *) calloc( DoF, sizeof( float ) );
final_thetas    = (float *) calloc( DoF, sizeof( float ) );
initial_speeds  = (float *) calloc( DoF, sizeof( float ) );
final_speeds    = (float *) calloc( DoF, sizeof( float ) );
initial_accels  = (float *) calloc( DoF, sizeof( float ) );
final_accels    = (float *) calloc( DoF, sizeof( float ) );
        /* Allocate memory to torque arrays */
torque_array = (float *) calloc( DoF, sizeof( float ) );
        /* Initialize PID ctrl data structure */
for( joint = 0; joint < DoF; joint++ )
        pid_data->prev_err_qq[joint]=pid_data->prev_err_ii[joint]=0.0;
        /* This will insure that any statement calling this code will
         * require at most 5 s to execute. Avoid infinite loops! */
mot_data->max_req_time = 5.0;
        /* Obtain start and end-point of joint configurations for current move
         * final_thetas = ideal solution angles computed by inverse kinematics.
         * initial_accels and final_accels = 0.0 for all joints.
         * This case corresponds to Case.A in thesis */
        /* Start and end-point position values */
initial_thetas[0] = offset_theta[0] = mot_data->prev_thetas[0] = 0.0 ;
final_thetas[0]   = -0.6 ;
initial_thetas[1] = offset_theta[1] = mot_data->prev_thetas[1] = -0.785;
final_thetas[1]   = -1.57 ;
initial_thetas[2] = offset_theta[2] = mot_data->prev_thetas[2] = -0.61 ;
final_thetas[2]   = 0.0 ;
        /* Start and end-point speeds and acceleration values */
for ( joint = 0; joint<DoF ; joint++)
        initial_speeds[joint]=final_speeds[joint]=
        initial_accels[0]=final_accels[0]=0.0 ;
printf(" Enter lambda dynamic avg_factor (0-1): \n");
```

```
scanf("%f", &avg_factor);
scanf("%c", &nl);       /* Discard newline char */
printf("\n");
        /* Set Dynamic parameters for start and end states
         * and calculate last control action and that of the initial state.
         * Initial state */
inertia_matrix( final_thetas, D);
centripetal_coriolis_and_gravity_matrix( final_thetas, final_speeds, HG);
calc_torque( mot_data->holding_torque, final_accels, D, HG );
inverse_inertia_matrix( D, D_inv );
calc_alpha(mot_data->alpha_final, D_inv );
calc_beta(mot_data->beta_final, D_inv, HG, mot_data->holding_torque);
        /* Final state */
inertia_matrix( initial_thetas, D);
inverse_inertia_matrix( D, D_inv );
centripetal_coriolis_and_gravity_matrix( initial_thetas, initial_speeds, HG);
calc_torque( mot_data->torque_last, initial_accels, D, HG );
calc_alpha(alpha, D_inv );
calc_beta(beta, D_inv, HG, mot_data->torque_last);

printf("crs_init() exited. \n");
} /* end crs_init() */

/*-------------------------------------------------------------------------
 * Procedure to actually control initialisation of DAQ boards
 *-------------------------------------------------------------------------*/
void crs_init_daq(void)
{
short   daqErr,joint;

for(joint = 1;joint <= DoF;joint++)
{
        daqErr = configure_ports(joint);
        if (daqErr)
            printf("Error #:%d Configuring device: %i\n", daqErr, joint);
        daqErr=counter_init_state(joint);
        if (daqErr)
            printf("Error #: %d Counter init state of device: %i\n",daqErr,joint);
        daqErr= clear_and_load_init_value(joint);
        if (daqErr)
            printf("Error #: %d  Clear and load init of device: %i\n",daqErr,joint);
}
} /* end crs_init_daq() */

/*-------------------------------------------------------------------------
 * Procedure to configure DAQ boards
 *-------------------------------------------------------------------------*/
short configure_ports(short device)
{
short daqErr;

        /* Config Port 0 (Port A in LabPC+) for reading: Data Input */
daqErr=DIG_Prt_Config (device, DPortA, 0, read);
        /* Config Port 1 (Port B in LabPC+) for reading: Data Input */
daqErr= daqErr || DIG_Prt_Config (device, DPortB, 0, read);
        /* Config Port 2 (Port C in LabPC+) for Writing: Control Output */
```

```
daqErr= daqErr || DIG_Prt_Config (device, DPortC, 0, write);
if (daqErr)
        printf(" Error Number: %d \n", daqErr);
return(daqErr);
} /* end configure_ports() */

/*-------------------------------------------------------------------------
 * Procedure to initialise PCB interface
 *-------------------------------------------------------------------------*/
short counter_init_state(short device)
{
short daqErr;

        /* "Count Up/Down" (DPortC-Line0) is set to high */
daqErr = DIG_Out_Line (device, DPortC, Line0, 1);
        /* The other "Count Down/Up" (DPortC-Line3) is also set to high */
daqErr = daqErr || DIG_Out_Line (device, DPortC, Line3, 1);
        /* "Clear" line initially to low */
daqErr = daqErr || DIG_Out_Line (device, DPortC, Line1, 0);
        /* "Load" line initially to high */
daqErr = daqErr || DIG_Out_Line (device, DPortC, Line2, 1);
if (daqErr)
        printf(" Error Number: %d \n", daqErr);
return(daqErr);
} /* end counter_init_state() */

/*-------------------------------------------------------------------------
 * Procedure to reset PCB interface
 *-------------------------------------------------------------------------*/
short clear_and_load_init_value(short device)
{
short daqErr;

        /* Clear everything (DPortC-Line1): send high pulse down "Clear" = Line1 */
daqErr = DIG_Out_Line (device, DPortC, Line1, 1);
daqErr = daqErr || DIG_Out_Line (device, DPortC, Line1, 0);
        /* Load  Initial Value : send low pulse down "Load" = Line2 */
daqErr = daqErr || DIG_Out_Line (device, DPortC, Line2, 0);
daqErr = daqErr || DIG_Out_Line (device, DPortC, Line2, 1);
        /* Reset analog output (DAC0) = 0 */
daqErr = daqErr || AO_VWrite (device, DAC0, 0 );
if (daqErr)
        printf(" Error Number: %d \n", daqErr);
return(daqErr);
} /* end clear_and_load_init_value() */

/*************************************************************************
 *                      Module dynamics.c                                *
 *                                                                       *
 *                      Jaime Valls Miro                                 *
 *************************************************************************/
#include<crs_defs.h>
extern  float   *final_thetas, *initial_thetas;

/*-------------------------------------------------------------------------
 * Procedure to calculate variable inertia matrix - D(q)
```

```
 *-------------------------------------------------------------------------*/
void inertia_matrix(float *thetasp, float (*D)[DoF] )
{
        /* Waist Joint, 0 */
D[0][0] = m1*pow(lm1,2) +
          (m2*pow(lm2,2)+(m3+m4+m5)*pow(lm3,2))*pow(cos(*(thetasp+1)),2)+
          (m4*pow(lm4,2)+m5*pow(lm5,2))*pow(cos(*(thetasp+2)),2)+
          2*lm3*(m4*lm4+m5*lm5)*cos(*(thetasp+1))*cos(*(thetasp+2))+
          Jmeff[0][0];
D[0][1]= 0 + Jmeff[0][1];
D[0][2]= 0 + Jmeff[0][2];
        /* Upper-arm Joint, 1 */
D[1][0] = D[0][1];
D[1][1] = m2*pow(lm2,2)+(m3+m4+m5)*pow(lm3,2) + Jmeff[1][1];
D[1][2] = -((m4*lm4+m5*lm5)*lm3*cos((*(thetasp+1))+(*(thetasp+2))))+Jmeff[1][2];
        /* Fore-arm Joint, 2 */
D[2][0] = D[0][2];
D[2][1] = D[1][2];
D[2][2] = m4*pow(lm4,2)+m5*pow(lm5,2) + Jmeff[2][2];
}  /* end inertia_matrix() */

/*-------------------------------------------------------------------------
 * Procedure to inverse a square 3x3 matrix - D(q)
 *-------------------------------------------------------------------------*/
void inverse_inertia_matrix(float (*D)[DoF], float (*D_inv)[DoF])
{
float   det_matrix, Adj[DoF][DoF], inv_det_matrix;
int     i,j;

        /* Compute matrix determinant */
det_matrix=(D[0][0]*D[1][1]*D[2][2]+D[1][0]*D[2][1]*D[0][2]+D[2][0]*D[0][1]*D[1][2])-
           (D[2][0]*D[1][1]*D[0][2]+D[2][1]*D[1][2]*D[0][0]+D[1][0]*D[0][1]*D[2][2]);
        /* Compute matrix adjoints */
Adj[0][0] = (D[1][1]*D[2][2])-(D[2][1]*D[1][2]);
Adj[0][1] = - ((D[1][0]*D[2][2])-(D[2][0]*D[1][2]));
Adj[0][2] = (D[1][0]*D[2][1])-(D[2][0]*D[1][1]);
Adj[1][0] = - ((D[0][1]*D[2][2])-(D[2][1]*D[0][2]));
Adj[1][1] = (D[0][0]*D[2][2])-(D[2][0]*D[0][2]);
Adj[1][2] = - ((D[0][0]*D[2][1])-(D[2][0]*D[0][1]));
Adj[2][0] = (D[0][1]*D[1][2])-(D[1][1]*D[0][2]);
Adj[2][1] = - ((D[0][0]*D[1][2])-(D[1][0]*D[0][2]));
Adj[2][2] = (D[0][0]*D[1][1])-(D[1][0]*D[0][1]);
inv_det_matrix = (1/det_matrix);
D_inv[0][0] = inv_det_matrix*Adj[0][0];
D_inv[0][1] = inv_det_matrix*Adj[1][0];
D_inv[0][2] = inv_det_matrix*Adj[2][0];
D_inv[1][0] = inv_det_matrix*Adj[0][1];
D_inv[1][1] = inv_det_matrix*Adj[1][1];
D_inv[1][2] = inv_det_matrix*Adj[2][1];
D_inv[2][0] = inv_det_matrix*Adj[0][2];
D_inv[2][1] = inv_det_matrix*Adj[1][2];
D_inv[2][2] = inv_det_matrix*Adj[2][2];
}  /* end inverse_inertia_matrix() */

/*-------------------------------------------------------------------------
 * Procedure to calculate centripetal and coriolis matrix - HG = H(q,qdot), G(q)
```

```
 *-------------------------------------------------------------------------*/
void centripetal_coriolis_and_gravity_matrix(float *thetasp,float *speedsp,float *HG)
{
float   H[DoF][DoF][DoF], G[DoF];

        /* Waist Joint, 0 */
H[0][0][1] = H[0][1][0] = - ( (m2*pow(lm2,2) +
             (m3+m4+m5)*pow(lm3,2))*sin(*(thetasp+1))*cos(*(thetasp+1)) +
             (m4*lm4+m5*lm5)*lm3*sin(*(thetasp+1))*cos(*(thetasp+2)) );
H[0][0][2] = H[0][2][0] = - ( (m4*pow(lm4,2)+
              m5*pow(lm5,2))*cos(*(thetasp+2))*sin(*(thetasp+2)) +
              (m4*lm4+m5*lm5)*lm3*cos(*(thetasp+1))*sin(*(thetasp+2)) );
HG[0] = H[0][0][1]*(*(speedsp+0))*(*(speedsp+1))+
        H[0][0][2]*(*(speedsp+0))*(*(speedsp+2))+
        H[0][1][0]*(*(speedsp+1))*(*(speedsp+0))+
        H[0][2][0]*(*(speedsp+2))*(*(speedsp+0))+
        0.5*(*(speedsp+0)) + 1.0*(-sign(*(speedsp+0)))*moving(*(speedsp+0));
        /* Upper-arm Joint, 1 */
H[1][0][0] = -H[0][0][1];
H[1][2][2] = (m4*lm4+m5*lm5)*lm3*sin((*(thetasp+1))+(*(thetasp+2)) );
G[1] = - (m2*lm2+(m3+m4+m5)*lm3)*cos(*(thetasp+1))*g;
HG[1] = H[1][0][0]*pow(*(speedsp+0),2)+
        H[1][2][2]*pow(*(speedsp+2),2)+
        G[1] +
        0.8*(*(speedsp+1)) + 1.0*(-sign(*(speedsp+1)))*moving(*(speedsp+1));
        /* Fore-arm Joint, 2 */
H[2][0][0] =(m4*pow(lm4,2)+m5*pow(lm5,2))*cos(*(thetasp+2))*sin(*(thetasp+2))+
             (m4*lm4+m5*lm5)*lm3*cos(*(thetasp+1))*sin(*(thetasp+2));
H[2][1][1] = (m4*lm4+m5*lm5)*lm3*sin((*(thetasp+1)) + (*(thetasp+2)));
G[2]=(m4*lm4+m5*lm5)*cos(*(thetasp+2))*g;
HG[2] = H[2][0][0]*pow(*(speedsp+0),2) +
        H[2][1][1]*pow(*(speedsp+1),2) +
        G[2] +
        0.3*(*(speedsp+2)) + 0.7*(-sign(*(speedsp+2)))*moving(*(speedsp+2));
}  /* end centripetal_coriolis_and_gravity_matrix() */

/*-------------------------------------------------------------------------
 * Procedure to calculate alpha dyn. coefficient
 *-------------------------------------------------------------------------*/
void calc_alpha( float *alpha, float (*D_inv)[DoF] )
{
int     joint;

for(joint = 0; joint < DoF; joint++)
    alpha[joint] = D_inv[joint][joint];
}  /* end calc_alpha() */

/*-------------------------------------------------------------------------
 * Procedure to calculate beta dyn. coefficient
 *-------------------------------------------------------------------------*/
void calc_beta( float *beta, float (*D_inv)[DoF], float *HG, float *torques )
{
int     i,j;

for(i = 0; i < DoF; i++)
{
```

```
    beta[i] = 0.0;
    for(j = 0; j < DoF ; j++)
    {
        beta[i]-= (D_inv[i][j]*HG[j]);
        if(i! = j)
            beta[i]+= (D_inv[i][j]*torques[j]);
    }
}
}  /* end calc_beta() */

/*-------------------------------------------------------------------------
 * Procedure to calculate joint torques for given state (thetas, speeds, accels)
 *-------------------------------------------------------------------------*/
void calc_torque(float *torques, float *accels, float (*D)[DoF], float *HG )
{
int     i,j;

for(i = 0; i < DoF ; i++)
{
    torques[i] = 0.0;
    for(j = 0; j < DoF ; j++)
         torques[i]+= (D[i][j]*accels[j]);
    torques[i]+= HG[i];
}
}  /* end calc_torques() */

/*************************************************************************
 *                    Module control.c                                   *
 *                                                                       *
 *                    Jaime Valls Miro                                   *
 *************************************************************************/
#include<crs_defs.h>
extern  float   torque_max[DoF],current_torques[samples][DoF],tot_time,*alpha,*beta;
extern  double  Vm[samples][DoF];
extern  int     i;

/*-------------------------------------------------------------------------
 * Procedure to calculate near-optimal-time torque-curves
 *-------------------------------------------------------------------------*/
void torque_curve(int joint, mot_data_struct *mot_data, float *final_thetas,
                  float *current_speeds, float *current_thetas)
{
float   gamma_plus,gamma_minus,temp_torque,temp_volts,temp_current,thetas_sbift[DoF],
        ptorque_max[DoF],mtorque_max[DoF],di;

        /* Calculate current maximum admissible torques (positive and negative) */
di=( mot_data->torque_last[joint] > 0.0 ) ? 0.0 :  (torque_max[joint] -
        mot_data->torque_last[joint])/(sample_time*Km*N*gear_eff[joint]);
ptorque_max[joint]=temp_torque = (max_motor_voltage -
        (*(current_speeds+joint))*N*Kbemf - di*L )*Km*N*gear_eff[joint]/Rm[joint];
ptorque_max[joint]=(temp_torque > torque_max[joint])?torque_max[joint]:temp_torque;
di=( mot_data->torque_last[joint] < 0.0 ) ? 0.0 : (-torque_max[joint] -
        mot_data->torque_last[joint])/(sample_time*Km*N*gear_eff[joint]);
mtorque_max[joint]=temp_torque = ( -max_motor_voltage -
        (*(current_speeds+joint))*N*Kbemf - di*L )*Km*N*gear_eff[joint]/Rm[joint];
mtorque_max[joint]=(temp_torque<-torque_max[joint])?-torque_max[joint]:temp_torque;
```

```
thetas_shift[joint]=(*(current_thetas+joint)) - final_thetas[joint];
gamma_plus=mot_data->alpha_avg[joint]*ptorque_max[joint]+mot_data->beta_avg[joint];
gamma_minus=mot_data->alpha_avg[joint]*mtorque_max[joint]+mot_data->beta_avg[joint];
if(*(current_speeds+joint) >= 0)
    if( thetas_shift[joint] < pow((*(current_speeds+joint)),2)/(2*gamma_minus) )
        mot_data->torque_last[joint] = ptorque_max[joint];
    else
        mot_data->torque_last[joint] = mtorque_max[joint];
else
    if( thetas_shift[joint] <= pow((*(current_speeds+joint)),2)/(2*gamma_plus) )
        mot_data->torque_last[joint] = ptorque_max[joint];
    else
        mot_data->torque_last[joint] = mtorque_max[joint];
} /* end torque_curve() */

/*-------------------------------------------------------------------------
 * Procedure to calculate PID control
 *-------------------------------------------------------------------------*/
float pid_ctrl( pid_ctrl_data_struct *pid_data,int joint,float *current_thetas,
                float *final_thetas)
{
float pid_error_torque;

        /* PID depending on type of PID control */
#if defined(PID)
        float   Kpp[DoF] = {200.0,200.0,200.0};
        float   Kii[DoF] = {0.0,0.0,20.0};
        float   Kdd[DoF] = {0.6,0.6,2.0};
#endif
#if defined(ComputerTorque)
        float           Kpp[DoF]={400.0,400.0,400.0};
        float   Kii[DoF]={0.0,0.0,0.0};
        float   Kdd[DoF]={25.0,25.0,25.0};
#endif
#if defined(FeedForward) || defined(FeedForward_NoModel)
        float   Kpp[DoF]={30.0,100.0,60.0};
        float   Kii[DoF]={0.0,0.0,0.0};
        float   Kdd[DoF]={2.0,1.5,2.0};
#endif
        /* Calculate control error terms.
         * Position control error term */
pid_data->err_qq[joint] = final_thetas[joint]-(*(current_thetas+joint));
        /* Velocity control error term */
pid_data->err_vv[joint]=(pid_data->err_qq[joint]-
        pid_data->prev_err_qq[joint])/sample_time;
        /* Integral control error term */
pid_data->err_ii[joint] = pid_data->prev_err_ii[joint] +
        (pid_data->err_qq[joint]+pid_data->prev_err_qq[joint])*(sample_time/2);
        /* Motor control torques using PID law */
pid_error_torque=Kpp[joint]*pid_data->err_qq[joint]
        +Kdd[joint]*pid_data->err_vv[joint]+Kii[joint]*pid_data->err_ii[joint];
        /* Save current pos. and integ. error for next iteration of PID control*/
pid_data->prev_err_qq[joint] = pid_data->err_qq[joint];
pid_data->prev_err_ii[joint] = pid_data->err_ii[joint];
return(pid_error_torque);
```

```
} /* end pid_ctrl() */

/*****************************************************************************
 *                      Module daq_ctrl.c                                    *
 *                                                                           *
 *                      Jaime Valls Miro                                     *
 *****************************************************************************/
#include<crs_defs.h>
extern  float   torque_max[DoF], Vm[samples][DoF],
                current_torques[samples][DoF], current_accels[samples][DoF];
extern  int     i;

/*---------------------------------------------------------------------------
 * Procedure to calculate Voltage control action corresponding to the control
 *  torque and send it down the DA (actuator model necessary)
 *--------------------------------------------------------------------------*/
float send_torque_control_voltage(float *torque, float *current_speeds,
                                  mot_data_struct *mot_data, short joint)
{
short   daqErr;
double  V;
float   temp_torque, di;

di=(*(torque+joint)-current_torques[i-1][joint])/(sample_time*Km*N*gear_eff[joint]);
V=Rm[joint]*(*(torque+joint))/(Km*N*gear_eff[joint])+ L*di +
         (*(current_speeds+joint))*N*Kbemf ;
         /* Pre-L.A. voltage, Ka=2, Op-Amp gain = 2 */
         /* Linear amplf. Good enough but could be improved */
V/=4.0;
V = (V>=0)? ((V >= max_PC_voltage ) ? max_PC_voltage : V):  /* Voltage saturation */
            ((V <= -max_PC_voltage ) ? - max_PC_voltage : V);
daqErr = AO_VWrite (joint+1, DAC0, V);
        /* The higher the gain, the flatter the responses of Im, but
         * also more steady-state error */
        /* Calculate last torque applied with newly calcul'd volt. control action */
temp_torque=(V*4-(*(current_speeds+joint))*N*Kbemf-di*L)*
Km*N*gear_eff[joint]/Rm[joint];
return(temp_torque);
} /* end send_torque_control_voltage() */

/*---------------------------------------------------------------------------
 * Procedure to send PID Voltage control action down the DA
 *              (No actuator model necessary)
 *--------------------------------------------------------------------------*/
void send_PID_control_voltage(float torque_error, float *current_speeds, short joint)
{
short   daqErr;
double  Vm;

Vm = torque_error;
/* Control voltage */
Vm = (Vm>=0)? ( (Vm >= max_motor_voltage ) ? max_motor_voltage  : Vm) :
              ((Vm <= - max_motor_voltage ) ? -max_motor_voltage  : Vm);
daqErr = AO_VWrite (joint+1, DAC0, Vm/4.001);
}/* end send_PID_control_voltage() */
```

Appendix C

PCB Schematics and Circuit Diagram

The schematics of the interface PCB designed for the control of the CRS A251 manipulator from a stand-alone PC are shown below. These were developed with the aid of the software package EASY-PC, while OrCAD software was employed in the design of the circuit diagram. The reader may refer to Sections 7.2 and 7.4 for more details about the design and implementation of the custom interface.

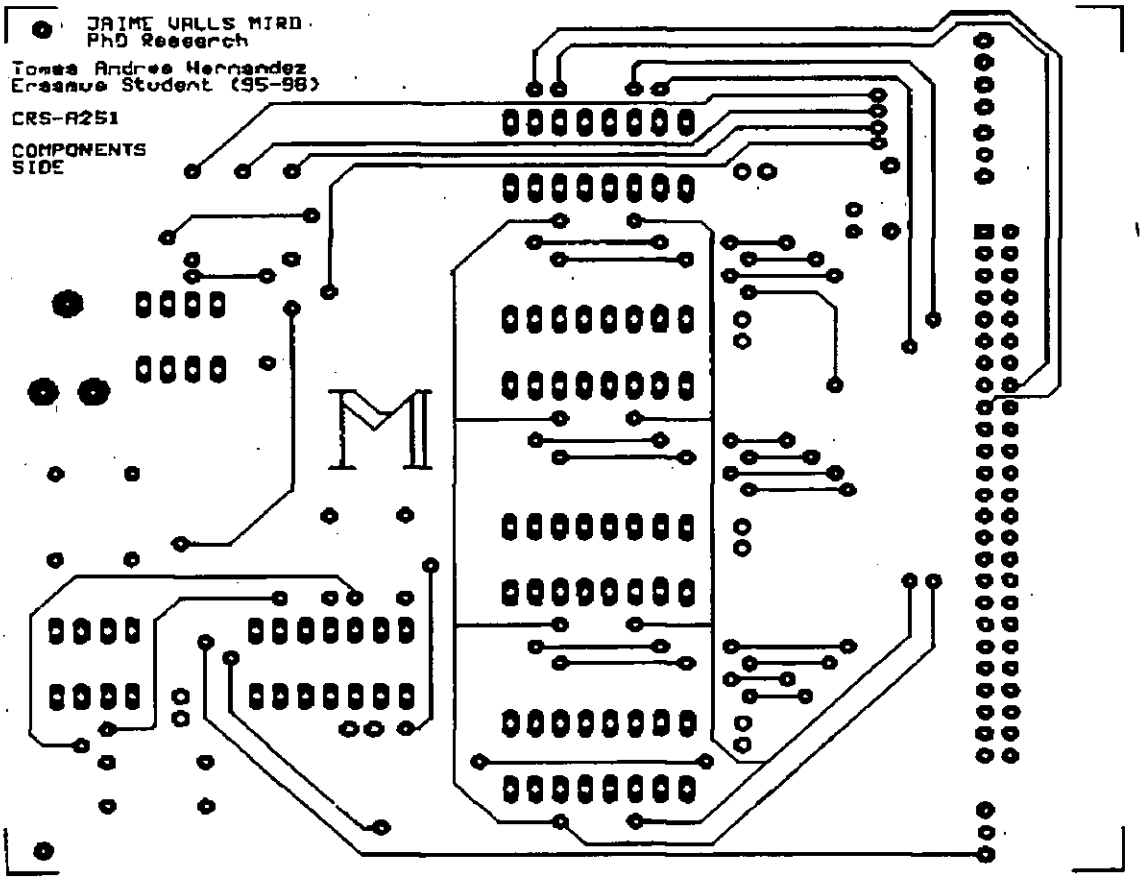

Figure C.1: Interface PCB schematic (top side).

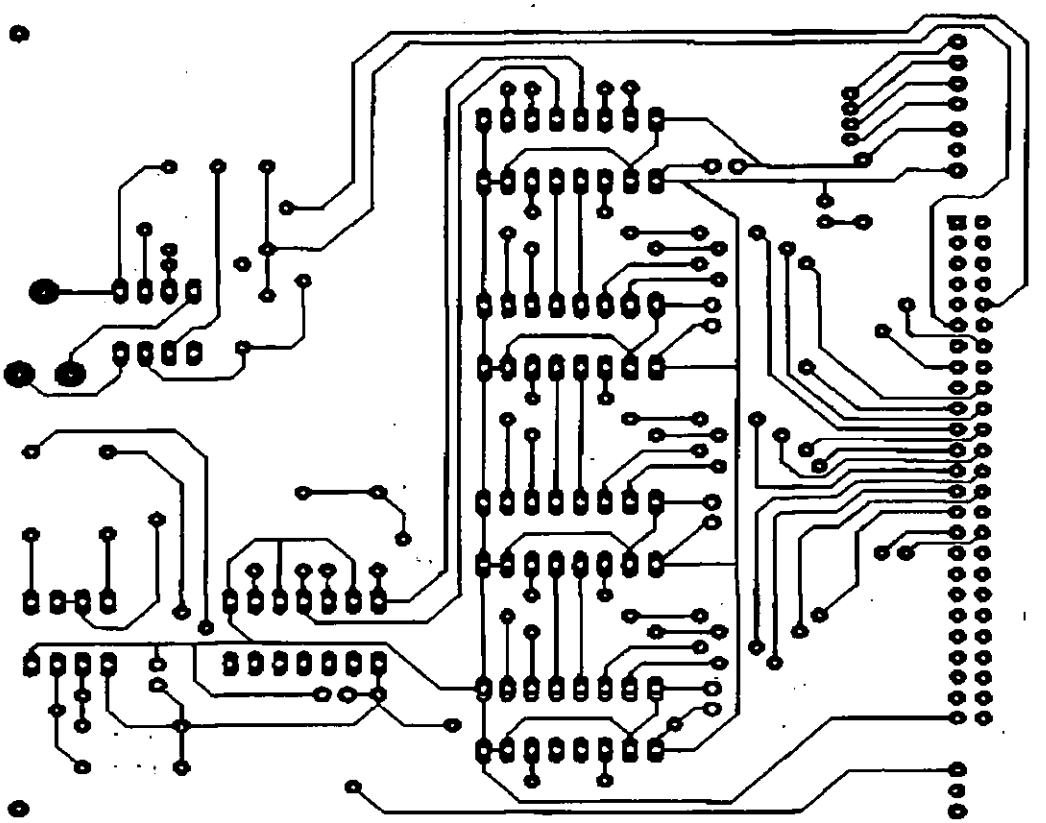

Figure C.2: Interface PCB schematic (bottom side).

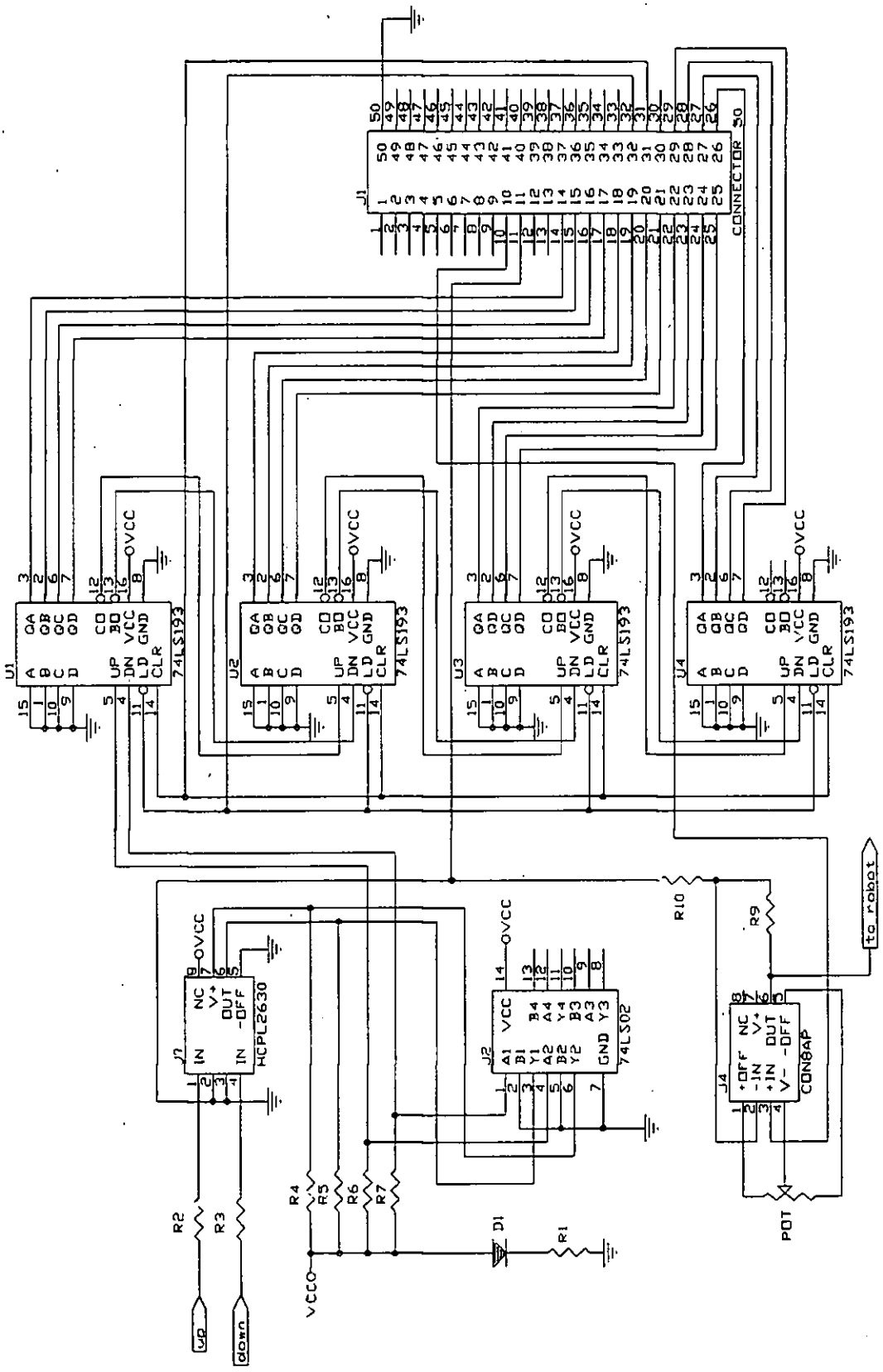

Figure C.3: Interface PCB circuit diagram.

Appendix D

Cubic Polynomial Validation Trajectories

The following curves represent the trajectory (position, velocity and acceleration) followed by the waist joint during the validation experiments described in Section 5.3.4. As both the upper-arm and fore-arm give similar looking profiles, they have not been included.

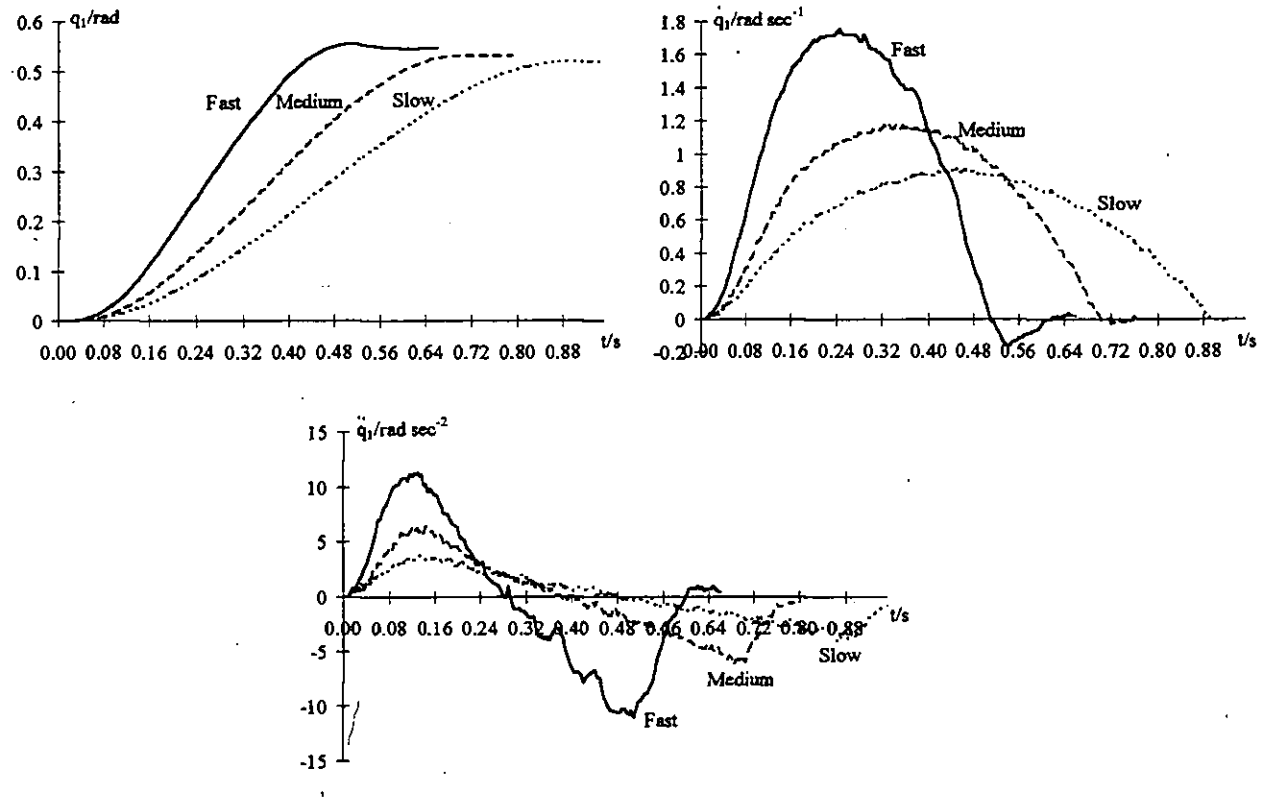

Figure D.1: Medium speed (0.7 s) cubic polynomial trajectory followed by Joint 1 during validation experiments.

Appendix E

Video

- *Investigations into an Optimal Approach for On-line Robot Trajectory Planning and Control*, Miró J.V., PhD thesis video, 1997.

Appendix F

Publications

1. *The Use of Computer Graphics for Robot Motion Simulation*, Miró J.V., Stoker M. and Gill R., Proceedings of the 11th ISPE/IEE/IFAC International Conference on CAD/CAM, Robotics & Factories of the Future, Pereira (Colombia), pp. 884-889, 1995.

2. *A Graphical Prototyping Approach for Robotic Simulation Environments*, Miró J.V., Stoker M. and Gill R., Proceedings of the Middlesex University Conference on Research in Technology, London (UK), Faculty of Technology technical report TR 1.96, pp. 32-36, 1995.

3. *Optimal Dynamic Trajectory-planner/Controller Applied to Industrial Manipulators*, Miró J.V., Stoker M., White A.S. and Gill R., Proceedings of the 12th ISPE/IEE/IMechE International Conference on CAD/CAM, Robotics & Factories of the Future", London (UK), pp. 180-186, 1996.

4. *On-line Time-optimal Algorithm for Manipulator Trajectory Planning*, Miró J.V., White A.S. and Gill R., Proceedings of the European Control Conference, Brussels (Belgium), Session TH-E G5, paper number 661 (in CD-ROM), 1997.

The Use of Computer Graphics for Robot Motion Simulation

Jaime Valls Miro, Mark Stoker, Raj Gill
Middlesex University
Bounds Green Road, London N11 2NQ, U.K.
Tel: 0181-362 5219, Fax: 0181-361 1726
Contact Email: jaimel@mdx.ac.uk.
http://www.mdx.ac.uk/www/ammc/ammc.html

Abstract

This paper describes how computer graphics can be integrated into the design, evaluation, off-line programming and real-time control of robotic systems. Robot parameters such as kinematics, dynamics and control are incorporated to produce accurate motion simulations. Three dimension CAD schematics of the manipulator and the working environment have allowed a number of concepts to be rapidly evaluated long before the expensive process of detailed design and manufacturing. Conventional CAD systems can then be used to carry out the designs which are in turn fed back to the graphics environment for validation. A case study is presented which describes the development of an automated workcell within the graphical simulation environment.

Once the manipulator design is established particular attention is given to the motion improvement that the use of robot dynamics may have in the early stages of the motion pipeline. Traditionally the dynamic model has only been accounted for during the control loop, and in most cases is totally ignored . A method is proposed where path planning can be enhanced by the use of the manipulator dynamics. Although Lagrange-Euler formulation was originally studied due to its closed form, more computationally suitable forms like the recursive Newton-Euler approach are being investigated.

Keywords: Computer Graphics, Simulation, Prototype, Manipulator Motion

1 Introduction

The purpose of systems study through modelling is to aid the analysis, understanding, design, operation, prediction and/or control of systems without actually constructing and operating the real process [1]. Models play the role of the real objects whose analysis by real experimentation could be expensive, risky, time-consuming or even physically impossible [2]. Simulation models [1] have traditionally been approached by textual-based computer simulation languages, both discrete (GPSS, SIMULA, etc.) and continuous (ACSL, CSMP, etc.) some of which provide at most the capability to plot some simulation results in a simple graphical environment. However the rapid development of computer hardware and graphics software during the last decade has added a new dimension to the practice of modelling and simulation.

It is generally accepted that humans can relatively easily assimilate complex information from pictorial images ("A picture says more than a thousand words" - Confucius). Undoubtedly colour graphics and animation is considered a highly desirable feature in understanding the dynamics of system behaviour via simulation software. Indeed, this is found particularly attractive in robotics [3].

A solid object can be represented in a computer aided design package (CAD) using primitive solids such as cubes, cones, wedges, spheres, etc. which are added, subtracted, cut, etc. to form desired shapes for the robot parts and its operating environment. These can then be fed into the graphical simulation package where further non-geometric attributes such as motion definition, joint limits and speeds, input/output, etc. are attached to the solid model of the manipulator and devices in its surroundings.

2 Problem Overview

Diagnostic radio-isotopes are used in the Nuclear medicine as markers within the body. They are attached to selected molecules (tracers) allowing the passage and distribution of these molecules to be traced in the body for the examination of a given area of the body - bone, lung, liver and heart to name but a few [4].

Regulations [5] require every employing authority to take all necessary steps to restrict, As Low As Reasonably Practicable (ALARP [6]), the extent to which employees and other persons are exposed to ionising radiation, and impose limits on the doses of ionising radiation which employees and other persons may receive in any calendar year. This project is an attempt to fulfill such requirements by designing an automated radiopharmaceutical dispenser to prepare precise individual patient prescriptions [7, 4, 8]. The system consists of as isolator cabinet containing:

- Specialised programmable stations (e.g. syringe filling station described in Section 3.1).
- Consumables (syringes, vials) and the corresponding lead-shielded containers.
- Radio-isotope generator.
- A general purpose 5DoF manipulator to transfer items between stations, remove lids, etc.

The system provides the required patient dose of radio-isotope delivered either in shielded syringes or vials.

Research is currently being carried out in the improvement that manipulator dynamic models may have in the early stages of the robot motion process, particularly in medical robotic workcells. The algorithms are being implemented and tested entirely within the same graphical simulation framework than previously used during the study of the robotic workcell layout. An overview is given at the end of the paper.

3 System Design Methodology

For the development of the system a suite of tools and technologies capable of matching the capabilities of the human user to the requirements demanded by the application was needed. Desktop virtual reality is an advance concept for the graphical design, prototyping and systems simulation which makes the designed objects' behaviours more accessible and understandable to the user. The attributes and associations between objects in a virtual environment permit an approximation to the nature and behaviour of such objects and processes which do not yet exist, thus providing the sort of front-end with which the user feels comfortable and accelerating the overall development process.

3.1 Virtual Prototyping

During the initial stages of the simulation process an accurate CAD solid model of the manipulator coupled with simple schematics of the rest of the parts in the workcell have allowed a number of different concepts to be rapidly evaluated. This has permitted important design principles to be established and verified at an early stage in the project and long before the expensive processes of detailed design and manufacture. For instance the robot datum-bracket [1] provided by the manufacturer was found far too bulky for the limited working space available and was quickly redesign from a stand position in the working plate into a wall-mounted home plate.

[1] Necessary to provide a repeatable position when robot uses incremental encoders

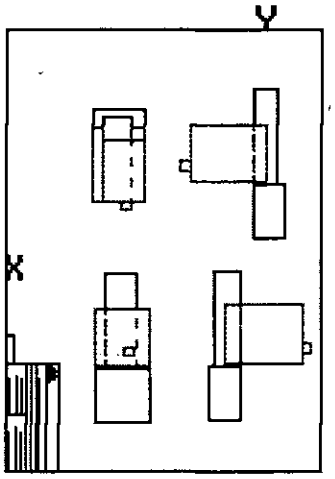

Figure 1: Syringe-filling-station 2D Prototype

Starting from 2D CAD designs as shown in figure 1, or directly from solid models of the components the prototypes are created. The ability to evaluate a mechanism for functional capability is critical at this stage because too much nonfunctional detail will impact dramatically the simulation performance. Hence, prototype modelling strategies must discriminate detail prudently, retaining component identity and functionality to comply with the conceptual design. This is explained with the following example.

Figure 2 represents the virtual prototype of the syringe-filling-station (s.f.s.) which was used throughout the project to test for reachability and collisions in the workcell as described below in section 3.2. The s.f.s. station is a 3 DoF mechanism used to transfer liquid to and from the syringes: 2 prismatic joints operate the syringe plunger and vial carriage to withdraw the liquid and a third rotational joint inverts the mechanism to perform this action. On the development of the prototype the main concern was to test for collisions with the environment when performing a rotational motion of the device. Since linear motion did not represent any constraint the prototype was designed with a single DoF (rotational) with the prismatic joints fully extended (worst case) to be tested against its surroundings.

After solid models of the required components have been developed, analytical work of building a system that clearly describes the operation of the workcell is started.

3.2 Workcell Layout. Path Planning

The first step in this process is to couple the necessary devices (those with any degree of motion) with kinematics, joint limits, speeds and accelerations, etc. Each device is then operationally positioned relative to the other devices describing the total system. A detailed path planning process is then carried out to test the layout against reachability and collision of the robot to fulfill all the tasks. Simulation through virtual reality made it possible to travel virtual distances within the workcell, highlight design flaws, illustrate the size of the facility, etc. thus allowing for a most practical environment to carry out the tasks.

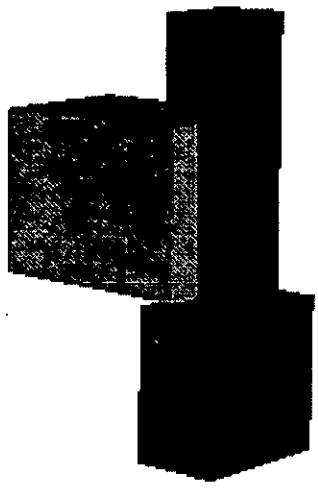

Figure 2: S.f.s. Solid Model Prototype

The initial simulation identified the following problems with the proposed workcell layout and allowed a correction to be posed.

- The enclosing cabinet design suggested by the manufacturer was found too narrow to allow for the motions of the robot and after consultations with the design engineers it was agreed to increase the depth of the cabinet model. These permitted a more flexible arrangement and path planning in the workcell.
- The lack of yaw motion in the robot (5 DoF) forced the use of a turn-over station for re-gripping the bottles carrying the radio-isotopes. The possibility of upgrading the manipulator to a 6 DoF was sought. After creating the model into the workcell it was found too large which restricted adequate motion planning within the workcell and the idea was scrapped.

Simple refinements like these would have been very laborious and expensive if a workcell mock-up was to be built. The approach described here allows for better conceptual implementation and process optimization when compared to traditional 2D or 3D wireframes and mechanical checking methods.

Once satisfied with the cell configuration and its performance, detailed CAD drawings are then developed, converted into solid models (Figure 3) and fed back to the graphics environment for validation and final visualization purposes, off-line programming and development of control and robot motion strategies.

4 Motion Improvement

Additional work is now being carried out to enhanced motion efficiency in the workcell by looking at the impact that the manipulator dynamic models may have during the path planning stage. Minimising current and torque transients in the manipulator actuators reduces some mechanical forces and stresses in the system [9], thus increasing the up-time of the robot and consequently the whole system. This is of particular importance for complex machinery in hazardous environments like the system described here.

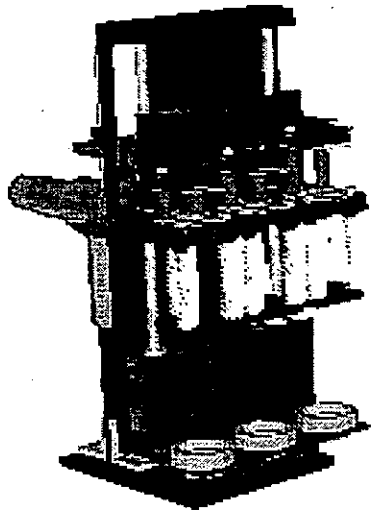

Figure 3: S.f.s. Final design

Having developed kinematic models for the robot - used during the workcell layout, work is now concentrated in obtaining an accurate dynamic model of the manipulator to optimize the paths that the robot must follow to perform a particular task. Tests are being carried out in the graphical simulation environment to calculate and visualize the torques required to move the robot at different positions, velocities and accelerations. Results are providing an insight into the dynamic needs of the manipulator. Paths and motion parameters will then be modified to allow for the robot to approach these configurations whenever possible during the execution of the tasks. In the limited working space that medical robotics have to offer this might not be a major improvement but work might be extrapolated to any robotic environment. In considering suitable algorithms to solve the inverse dynamic problem Lagrange-Euler [10, 11] turned out to be the most suitable option to gain a good understanding of the problem given its closed form. However more computationally attractive alternatives are under evaluation. In particular the recursive Newton-Euler algorithm [10] of which a parallel implementation on transputers is currently being developed.

5 Conclusions

This paper has described how advanced graphical simulation tools have been applied to benefit the total development cycle of an automated radiopharmaceutical dispenser. It would have been very difficult if not impossible, to lay out the workcell manually as the selected robot was hard pressed to reach the required target points in the workcell without collision. It has been shown how virtual prototyping and robot motion simulation lends itself to continuous designer-manufacturer participation, which results in a high degree of cooperation prior to actually building the hardware system. Graphical programming has emerged as the natural way to plan complex robot motions safely, quickly, and easily. Finally, an overview of current research in robot motion improvement by considering manipulator dynamics has been stated. At present, the prototype workcell is being assembled using the layout generated from the simulated workcell.

6 Acknowledgments

Jaime Valls Miro is currently undertaking a PhD in the area of robot motion and computer simulation at Middlesex University and his research programs is being sponsored by British Nuclear Fuels Plc.

References

[1] Neelamkavil F., *Computer Simulation and Modelling*, John Wiley & Sons Ltd., Essex, UK 1987.

[2] Matko D., Karba R., Zupančič B., *Simulation and Modelling of Continuous Systems. A Case Study Approach*, Prentice Hall Int. Ltd., Hartfordshire, UK 1992.

[3] Smith L.A., Barnes S.A., "The Use of Computer Graphics to Assist Real-Time Manipulator Operation", In *Euriscon'94, European Robotics and Intelligent Systems Conf. Proceedings*, Vol 2, pp 637-646, 1994.

[4] Maclean R.,"An Analysis of the Possibilities for Automating 99^{m}Tc Dispensing", *MSc Report*, School of Mechanical and Manufacturing Engineering, Middlesex University, UK 1991.

[5] "Radiation Protection in Nuclear Medicine and Pathology", *Report No. 69*, The Institute of Physical Sciences in Medicine, UK 1991.

[6] Elliot, A. T., Murray, T., Hilditch, T.E., "Automated Radiopharmaceutical Dispensing", *Nuclear Medicine Communications*, Vol 9, pp 363-367, 1988.

[7] Majkowsky I., "Radiation doserate mapping of an Automated Radiopharmaceutical Dispenser", *MSc Report*, School of Mechanical and Manufacturing Engineering, Middlesex University, UK 1993.

[8] Stoker M., Solanki K., Gill R., Hardie R., Snoback R., "Hamilton Dispenser - A Complete Automated Radiopharmaceutical Dispenser (HAPD)", In *European Association of Nuclear Medicine Congress Proceedings*, Laussane, Switzerland 1993.

[9] Sanders D.A., *Making Complex Machinery Move - automatic programming and motion planning*, Research Studies Press Ltd., Somerset, UK 1993.

[10] Fu K.S., Gonzalez R.C., Lee C.S.G., *Robotics. Control, Sensing, Vision, and Intelligence*, McGraw-Hill Int. Editions, Singapore 1987.

[11] Paul R.P., *Robot Manipulators. Mathematics, Programming, and Control*, The MIT Press, Cambridge (MA), USA 1981.

A Graphical Prototyping Approach for Robotic Simulation Environments

Jaime Valls Miro, Mark Stoker, Raj Gill
Middlesex University
Bounds Green Road, London N11 2NQ, U.K.
Tel: 0181-362 5219, Fax: 0181-361 1726
Email: jaime1@mdx.ac.uk
http://www.mdx.ac.uk/www/ammc/ammc.html

Abstract

This paper describes how computer graphics can be integrated throughout the design stages of a robotic system. Three dimension CAD schematics of the manipulator and the working environment have allowed a number of concepts to be rapidly evaluated long before the expensive process of detailed design and manufacturing. Conventional CAD systems can then be used to carry out the designs which are in turn fed back to the graphic environment for validation. Robot parameters such as kinematics, dynamics and control are incorporated to produce accurate motion simulations. A case study is presented which describes the development of an automated workcell within the graphical simulation environment.

Keywords: Computer Graphics, Simulation, Prototype, Manipulator Motion

1 Introduction

The purpose of systems study through modelling is to aid the analysis, understanding, design, operation, prediction and/or control of systems without actually constructing and operating the real process [1]. Models play the role of the real objects whose analysis by real experimentation could be expensive, risky, time-consuming or even physically impossible [2]. Simulation models [1] have traditionally been approached by textual-based computer simulation languages, both discrete (GPSS, SIMULA, etc.) and continuous (ACSL, CSMP, etc.) some of which provide at most the capability to plot some simulation results in a simple graphical environment. However the rapid development of computer hardware and graphics software during the last decade has added a new dimension to the practice of modelling and simulation.

It is generally accepted that humans can relatively easily assimilate complex information from pictorial images ("A picture says more than a thousand words" - Confucius). Undoubtedly colour graphics and animation is considered a highly desirable feature in understanding the dynamics of system behaviour via simulation software. Indeed, this is found particularly attractive in robotics [3].

A solid object can be represented in a computer aided design package (CAD) using primitive solids such as cubes, cones, wedges, spheres, etc. which are added, subtracted, cut, etc. to form desired shapes for the robot parts and its operating environment. These can then be fed into the graphical simulation package where further non-geometric attributes such as motion definition, joint limits and speeds, input/output, etc. are attached to the solid model of the manipulator and devices in its surroundings.

2 Problem Overview

Diagnostic radio-isotopes are used in the Nuclear medicine as markers within the body. They are attached to selected molecules (tracers) allowing the passage and distribution of these molecules to be traced in the body for the examination of a given area of the body - bone, lung, liver and heart to name but a few [4].

Regulations [5] require every employing authority to take all necessary steps to restrict, As Low As Reasonably Practicable (ALARP [6]), the extent to which employees and other persons are exposed to ionising radiation, and impose limits on the doses of ionising radiation which employees

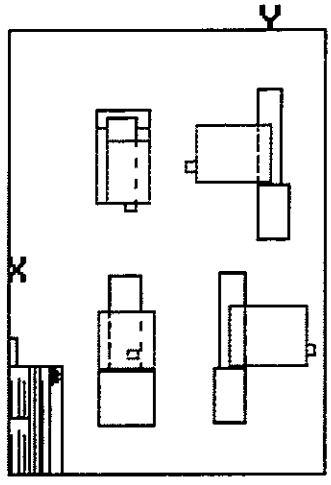

Figure 1: Syringe-filling-station 2D Prototype

and other persons may receive in any calendar year. This project is an attempt to fulfill such requirements by designing an automated radiopharmaceutical dispenser to prepare precise individual patient prescriptions [7, 4, 8]. The system consists of as isolator cabinet containing:

- Specialised programmable stations (e.g. syringe filling station described in Section 3.1).
- Consumables (syringes, vials) and the corresponding lead-shielded containers.
- Radio-isotope generator.
- A general purpose 5DoF manipulator to transfer items between stations, remove lids, etc.

The system provides the required patient dose of radio-isotope delivered either in shielded syringes or vials.

3 System Design Methodology

For the development of the system a suite of tools and technologies capable of matching the capabilities of the human user to the requirements demanded by the application was needed. Desktop virtual reality is an advance concept for the graphical design, prototyping and systems simulation which makes the designed objects' behaviours more accessible and understandable to the user. The attributes and associations between objects in a virtual environment permit an approximation to the nature and behaviour of such objects and processes which do not yet exist, thus providing the sort of front-end with which the user feels comfortable and accelerating the overall development process.

3.1 Virtual Prototyping

During the initial stages of the simulation process an accurate CAD solid model of the manipulator coupled with simple schematics of the rest of the parts in the workcell have allowed a number of different concepts to be rapidly evaluated. This has permitted important design principles to he established and verified at an early stage in the project and long before the expensive processes of detailed design and manufacture. For instance the robot datum-bracket [1] provided by the manufacturer was found far too bulky for the limited working space available and was quickly redesign from a stand position in the working plate into a wall-mounted home plate.

[1] Necessary to provide a repeatable position when robot uses incremental encoders

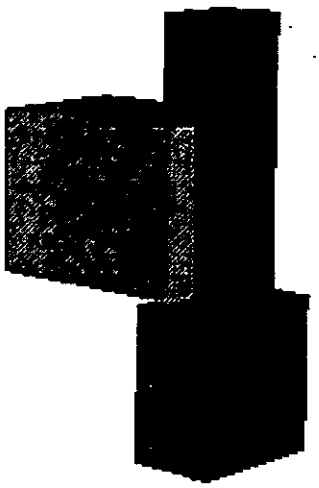

Figure 2: S.f.s. Solid Model Prototype

Starting from 2D CAD designs as shown in figure 1, or directly from solid models of the components the prototypes are created. The ability to evaluate a mechanism for functional capability is critical at this stage because too much nonfunctional detail will impact dramatically the simulation performance. Hence, prototype modelling strategies must discriminate detail prudently, retaining component identity and functionality to comply with the conceptual design. This is explained with the following example.

Figure 2 represents the virtual prototype of the syringe-filling-station (s.f.s.) which was used throughout the project to test for reachability and collisions in the workcell as described below in section 3.2. The s.f.s. station is a 3 DoF mechanism used to transfer liquid to and from the syringes: 2 prismatic joints operate the syringe plunger and vial carriage to withdraw the liquid and a third rotational joint inverts the mechanism to perform this action. On the development of the prototype the main concern was to test for collisions with the environment when performing a rotational motion of the device. Since linear motion did not represent any constraint the prototype was designed with a single DoF (rotational) with the prismatic joints fully extended (worst case) to be tested against its surroundings.

After solid models of the required components have been developed, analytical work of building a system that clearly describes the operation of the workcell is started.

3.2 Workcell Layout. Path Planning

The first step in this process is to couple the necessary devices (those with any degree of motion) with kinematics, joint limits, speeds and accelerations, etc. Each device is then operationally positioned relative to the other devices describing the total system. A detailed path planning process is then carried out to test the layout against reachability and collision of the robot to fulfill all the tasks. Simulation through virtual reality made it possible to travel virtual distances within the workcell, highlight design flaws, illustrate the size of the facility, etc. thus allowing for a most practical environment to carry out the tasks.

The initial simulation identified the following problems with the proposed workcell layout and allowed a correction to be posed.

- The enclosing cabinet design suggested by the manufacturer was found too narrow to allow for the motions of the robot and after consultations with the design engineers it was agreed to increase the depth of the cabinet model. These permitted a more flexible arrangement and path planning in the workcell.

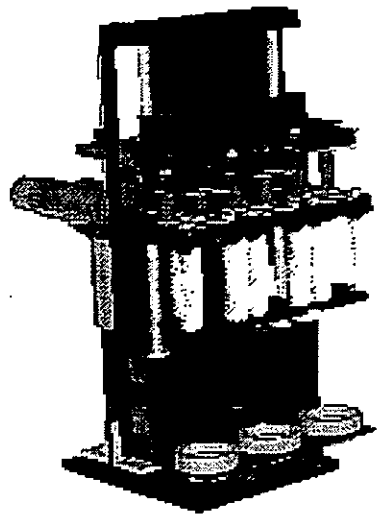

Figure 3: S.f.s. Final design

- The lack of yaw motion in the robot (5 DoF) forced the use of a turn-over station for re-gripping the bottles carrying the radio-isotopes. The possibility of upgrading the manipulator to a 6 DoF was sought. After creating the model into the workcell it was found too large which restricted adequate motion planning within the workcell and the idea was scrapped.

Simple refinements like these would have been very laborious and expensive if a workcell mock-up was to be built. The approach described here allows for better conceptual implementation and process optimization when compared to traditional 2D or 3D wireframes and mechanical checking methods.

Once satisfied with the cell configuration and its performance, detailed CAD drawings are then developed, converted into solid models (Figure 3) and fed back to the graphics environment for validation and final visualization purposes, off-line programming and development of control and robot motion strategies [12].

4 Conclusions

This paper has described how advanced graphical simulation tools have been applied to benefit the total development cycle of an automated radiopharmaceutical dispenser. It would have been very difficult if not impossible, to lay out the workcell manually as the selected robot was hard pressed to reach the required target points in the workcell without collision. It has been shown how virtual prototyping and robot motion simulation lends itself to continuous designer-manufacturer participation, which results in a high degree of cooperation prior to actually building the hardware system. Graphical programming has emerged as the natural way to plan complex robot motions safely, quickly, and easily.

At present, the prototype workcell is being assembled using the layout generated from the simulated workcell.

5 Acknowledgments

Jaime Valls Miro is currently undertaking a PhD in the area of robot motion and computer simulation at Middlesex University and his research programs is being sponsored by British Nuclear Fuels Plc.

References

[1] Neelamkavil F., *Computer Simulation and Modelling*, John Wiley & Sons Ltd., Essex, UK 1987.

[2] Matko D., Karba R., Zupančič B., *Simulation and Modelling of Continuous Systems. A Case Study Approach*, Prentice Hall Int. Ltd., Hartfordshire, UK 1992.

[3] Smith L.A., Barnes S.A., "The Use of Computer Graphics to Assist Real-Time Manipulator Operation", In *Euriscon'94, European Robotics and Intelligent Systems Conf. Proceedings*, Vol 2, pp 637-646, 1994.

[4] Maclean R.,"An Analysis of the Possibilities for Automating 99^{m}Tc Dispensing", *MSc Report*, School of Mechanical and Manufacturing Engineering, Middlesex University, UK 1991.

[5] "Radiation Protection in Nuclear Medicine and Pathology", *Report No. 63*, The Institute of Physical Sciences in Medicine, UK 1991.

[6] Elliot, A. T., Murray, T., Hilditch, T.E., "Automated Radiopharmaceutical Dispensing", *Nuclear Medicine Communications*, Vol 9, pp 363-367, 1988.

[7] Majkowsky I., "Radiation doserate mapping of an Automated Radiopharmaceutical Dispenser", *MSc Report*, School of Mechanical and Manufacturing Engineering, Middlesex University, UK 1993.

[8] Stoker M., Solanki K., Gill R., Hardie R., Snohack R., "Hamilton Dispenser - A Complete Automated Radiopharmaceutical Dispenser (HAPD)", In *European Association of Nuclear Medicine Congress Proceedings*, Laussane, Switzerland 1993.

[9] Sanders D.A., *Making Complex Machinery Move - automatic programming and motion planning*, Research Studies Press Ltd., Somerset, UK 1993.

[10] Fu K.S., Gonzalez R.C., Lee C.S.G., *Robotics. Control, Sensing, Vision, and Intelligence*, McGraw-Hill Int. Editions, Singapore 1987.

[11] Paul R.P., *Robot Manipulators. Mathematics, Programming, and Control*, The MIT Press, Cambridge (MA), USA 1981.

[12] Miro J.V., Stoker M., Gill R., "The Use of Computer Graphics for Robot Motion Simulation", In *The 11th ISPE/IEE/IFAC Int. Conf. on CAD/CAM, Robotics, and Factories of the Future Proc.*, Accepted and awaiting publication.

OPTIMAL DYNAMIC TRAJECTORY-PLANNER/CONTROLLER APPLIED TO INDUSTRIAL MANIPULATORS

J. V. Miró, M. Stoker, A. S. White and R. Gill
Middlesex University, Advanced Manufacturing and Mechatronics Centre,
Bounds Green Road, London N11 2NQ, U.K.
Contact email: jaime1@mdx.ac.uk
URL: http://www.mdx.ac.uk/www/ammc/ammc.html

Most current industrial manipulators address the problem of point-to-point motion in two sequential steps, namely trajectory planning and control tracking. Conventionally, the former stage is entirely based on kinematic considerations, hence with a strong geometric flavour, whereas linear feedback control laws are widely adopted later. However the dynamic behaviour of a mechanical manipulator is described by strongly non-linear differential equations. This paper describes a procedure which takes into account these non-linearities and joint coupling in an attempt to plan relistic minimum-time robot motions. A trajectory planner-controller of an industrial manipulator (CRS A251) based on optimal control theory has been simulated and shows that maximising the capabilities of the device can lead to an improvement in the manipulator time response of 25-35%.

Introduction

It is generally desirable to design the trajectory that a manipulator must follow as a smooth function of time, i.e., one which is continuous and ideally has a continuous first derivative too. Traditional trajectory planning strategies like cubic splines, linear functions with parabolic blends, critically damped, bang-bang, etc are entirely based on kinematic considerations to satisfy the set of constraints on the position, velocity and acceleration of the arm at a number of points along the desired trajectory. Regarding control strategies, practically all industrial manipulators currently in use are based on classical linear control theory such as PD or PID feedback control, etc... However, it is well known that mechanical manipulators are multibody systems whose dynamic behaviour is described by strongly non-linear differential equations. Non-linearities are associated both with position and velocity variables, and also their payloads. Although a few more advanced model-based industrial controllers - An C H & Atkeson C G & Hollerbach J M (1988) - compensate for some of the position-dependent non-linear terms such as gravity terms they all neglect the velocity-dependent terms in the controller design, which restricts the manipulator to slow motion if an accurate and smooth tracking is desired.

Both problems, trajectory planning and control are tightly coupled and yet traditionally have been regarded as separate problems which as stated above are approached sequentially. This often results in mathematically tractable solutions which do not maximise the performance that the manipulator is capable of at any given time. Both maximum speeds and accelerations and decelerations are limited for a

given robot structure by the torque capacity of the joint motor actuators and, accordingly, vary across the workspace. However, when a trajectory is planned a maximum speed and acceleration along each degree of freedom (DoF) is assumed. But in order not to exceed the actual capabilities of the device, these specifications must be chosen conservatively, possibly forcing the robot to be underutilised, e.g., the robot may be driven more slowly than necessary.

A procedure to overcome these drawbacks is proposed in this paper by the design of a trajectory planner-controller in configuration space which operates the manipulators close to its maximum efficiency between points, thus optimising motion time. This is addressed by the use of time optimal control theory based on Pontryagin's maximum principle - Kirk D (1970) - with special consideration to the manipulator dynamic equations of motion and drive torque bounds. The approach taken is similar to that adopted by some researchers in the past - Chen Y & Chien S Y P (1992) - who focused their work on time-optimal tracking of robotic manipulators along specified geometric paths. In this work, however, the actual trajectory the robot must follow is also considered for a realistic optimisation.

The validity of the trajectory planning algorithm presented in this paper has been established by numerical simulation of the first three joints of a five rigid links industrial manipulator, the CRS A251.

Problem Statement

The problem can then be summarized as follows: Using a closed form of manipulator dynamics [1] as shown in equation (1) find the optimal path that the robot tool centre point (TCP) should follow to move (in joint space) from the initial position q_{t_i} to the final position q_{t_f} while minimizing a performance index $J(t)$.

$$D(q)\ddot{q}+H(q,\dot{q})+G(q)=\tau \tag{1}$$

where $D(q)$ denotes the n x n inertia matrix (n = number of DoF of the manipulator), $H(q,\dot{q})$ is a n x 1 vector containing all velocity dependent terms arising from centripetal and coriolis forces and $G(q)$ represents the n x 1 gravity force terms. Joint torques are included in τ.

Depending on what parameters are considered in the cost function $J(t)$ the problem can be seen as an energy optimization problem, minimum time problem, a combination of both, acceleration optimization problem, etc...Efforts are being concentrated here into reducing the time required by the manipulator to achieve the desired location, i.e., the optimal time problem. Thus, the performance index can be expressed as:

$$J(t) = \int_{t_i}^{t_f} \sum_{i=1}^{n} 1 dt \tag{2}$$

Non-linear Optimal Trajectory Planner

If the terms of equation1 were expanded it would be easy to see that they are inherently nonlinear, both in position and velocity variables. This makes it impossible

[1] derived according to the Lagrangian formulation of mechanics. Friction is neglected without loss of generality

to design a trajectory planner-controller based on linear control theory. However, since equation1 represents the nonlinear plant dynamics correctly, we can use this model to actively compensate for all nonlinearities. To this end the problem was initially attempted by using modern optimal control theory - Kirk D (1970). Hence, a $2n$-dimensional state vector $y(t)=[y_1(t), y_2(t)]^T=[q, \dot{q}]^T$ will be used to rewrite the dynamic equation (1) in state-space form as

$$\dot{y}=A(y)+B(y)\tau \tag{3}$$

where

$$A(y)=\begin{bmatrix} y_2 \\ -D^{-1}(y_1)[H(y_1,y_2)+G(y_1)] \end{bmatrix} \quad B(y)=\begin{bmatrix} 0 \\ D^{-1}(y_1) \end{bmatrix} \tag{4}$$

Each joint of an industrial manipulator is driven by separate actuators and therefore the optimal trajectory is designed under constraints imposed on the maximum torque bounds that can be exerted by the actuators, i.e., $\tau \leq |\tau^{\pm}|$ which as shown do not depend on system state. Given the initial and final states y_{t_i} and y_{t_f} this results in a non-linear two-point boundary-value problem whose solution is very difficult, if not impossible to obtain - Kim B K & Shin K G (1985). In order to overcome such difficulty an alternative approach is proposed where the system equations are linearize and decoupled in terms of the joint variable q.

Linear Optimal Trajectory Planner

The dynamic model (3) can be first rewritten for each individual axis y_i. The 2-dimensional state vector for each axis can be now described as follows

$$\dot{y_i}=A(y)+B(y)\tau \tag{5}$$

In order to obtain a closed-form optimal solution to the problem we can further expand equation (5) into a form where there is a clear structure of the dependency between the acceleration of joint i and the actuator torque driving the joint

$$\dot{y_{2_i}} = \alpha_i(y_1)\tau_i + \beta_i(y_1, y_2, \tau) \qquad \text{for } i = 1 \ldots n \tag{6}$$

where

$$\begin{aligned} \alpha_i(y_1) &= D_{ii}^{-1}(y_1) \\ \beta_i(y_1, y_2, \tau) &= \sum_{j=1, j\neq i}^{n} D_{ij}^{-1}(y_1)\tau_j - \sum_{j=1}^{n} D_{ij}^{-1}(y_1)[H_j(y_1, y_2) + G_j(y_1)] \end{aligned}$$

D_{ij}^{-1} denotes the (i,j)th element of the inverse of $D(y_1)$. Coefficients $\alpha_i(y_1)$ and $\beta_i(y_1, y_2, \tau)$ are time-variant non-linear functions of the manipulator position, velocity and input. However, equation (6) could be regarded as an uncoupled linear system by assuming α_i and β_i constant during each sample interval ΔT at which the trajectory is generated. It will be assumed - Kirk D (1970) - that the sampling interval is small enough so that the continuous control signal can be approximated by a piecewise-constant function that changes only at each sample interval. That will

allow to approximate the inertial coupling in the joint at time t_c with the last control input at time $t_c - \Delta T$ with a minimal overall error - Kim B K & Shin K G (1985). Non-linearities on y_1 and y_2 are considered in feedback form by using the manipulator present state at the beginning of the current sampling period t_c, which is of particular interest in a real implementation. The resulting linear and decoupled equation for each link can be then described as

$$\dot{y_{2_i}} = \alpha_i \tau_i + \beta_i \qquad \text{for } i = 1 \ldots n \tag{7}$$

which reduces the problem to the solution of a quasi-double integrator problem - Kirk D (1970) - that must be solved in real-time at each sample interval. However, the inclusion of the state-dependent manipulator dynamics makes the system time-invariant no longer, proving then necessary to update the optimal switching curve at each sample time. Unlike the double integrator problem, the optimal switching curve at each time t can not be described by a unique equation, and therefore the control law needs to be considered for two separate cases depending on the current state:

$$\begin{cases} \text{if } y_{2_i}(t) \geq 0 \\ \quad \tau_i(t) = \begin{cases} \tau_i^+(t) & \text{if } y_{1_i}(t) < \dfrac{y_{2_i}^2(t)}{2\alpha_i(t)\tau_i^-(t) + \beta_i(t)} \\ \tau_i^-(t) & \text{otherwise} \end{cases} \\ \text{if } y_{2_i}(t) < 0 \\ \quad \tau_i(t) = \begin{cases} \tau_i^+(t) & \text{if } y_{1_i}(t) \leq \dfrac{y_{2_i}^2(t)}{2\alpha_i(t)\tau_i^+(t) + \beta_i(t)} \\ \tau_i^-(t) & \text{otherwise} \end{cases} \end{cases} \qquad \text{for } i = 1 \ldots n \tag{8}$$

The behaviour of such control law at any instant time t can be depicted by the optimal state switching curve of figure 1 where the two cases are clearly identifiable by the non-symmetry of the optimal curve.

Since the dynamic model is updated with state feedback information at the beginning of each interval ΔT the approximation error derived from the linearisation assuption is implicitly compensated. Furthermore, at present time t_c the optimal control input should be determined based on the dynamic behaviour of the manipulator over the period $[t_c, t_f]$. Yet the dynamic coefficients are known for t_c and t_f but not for the period in between. It is therefore necessary to find a way to describe the overall dynamic behaviour of the system for the remaining of the actual motion on the basis of the current state and the final state. Some methods have been devised - Kim B K & Shin K G (1985) - which propose an arithmetic average of the dynamic coefficients. Particularly simple is a zero order hold. A more general form implemented in Wiens G & Berggren M J (1991) has been adopted in which the overall dynamic behaviour of the manipulator is defined by two factors, one for each dynamic coefficient. The use of a factor allows for each boundary condition's dynamic performance to be weighted separately in the estimated final value.

Structure of the Optimal Trajectory Planner-Controller

The resulting algorithm that yields the minimum-time trajectory and control action the robot should follow to move from the initial to the final state can be formulated as follows:

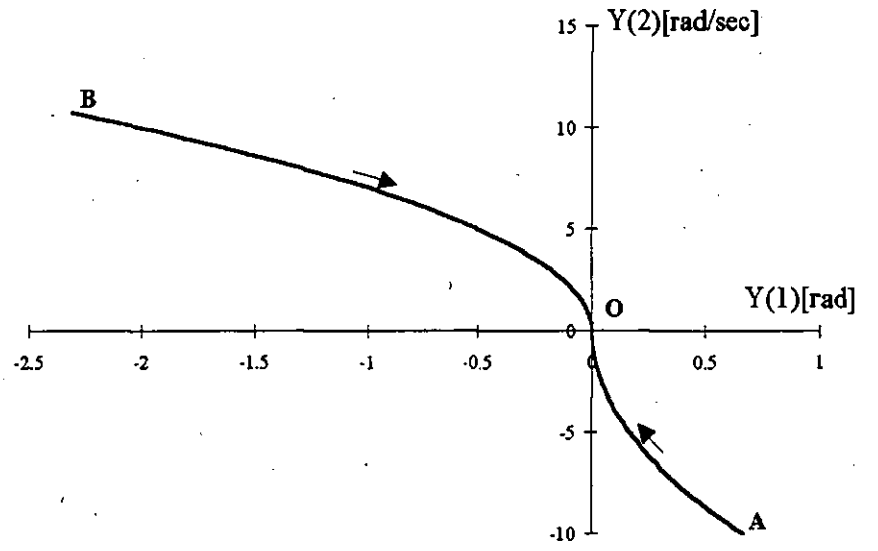

Figure 1: Quasi-Double Integrator State Switching Curve.

1. Derive the manipulator dynamic model. Lagrange mechanics has been used here, but other alternatives such as discrete dynamics, Newton-Euler or variations of Lagrange-Euler are viable also.

2. Given initial and final states compute the corresponding control actions from equation (1). The initial control action will act as the estimated control input into the algorithm.

3. Calculate dynamic behaviour for current state as described by equation (7). The linear approximation described by (7) is also applied now.

4. Average the dynamic coefficients to achieve an overall dynamic performance of the manipulator representative of the whole motion.

5. Use the current estimated dynamic model to update the switching curves and compute the optimal control according to (8).

6. Go back to point 3 if final state has not been reached.

A well-known problem with any bang-bang method is control chattering in the vicinity of the target state caused by frequent switchings of the control input. In the work presented here the possibility of applying feedforward torque control has been sought with successful results. This means that the actuator driving torque is solely determined by the current dynamic behaviour of the manipulator. Therefore point 6 of the algorithm is further extended to compare current state with target state. If the difference is less than some state bounds specified by the user, the control switches into a feedforward control.

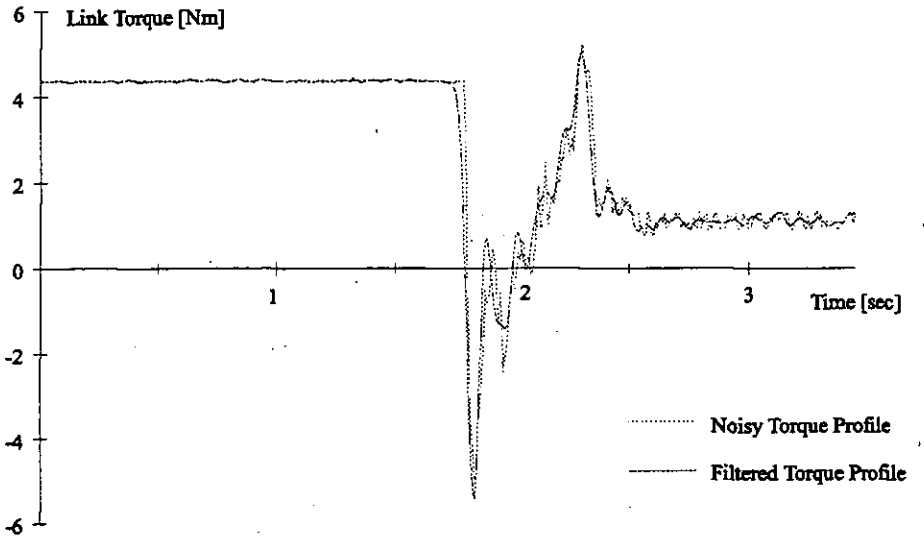

Figure 2: Measured PID torque response of robot joint.

Simulation Results

It was known that the manipulator controller follows a spline joint interpolated trajectory between points followed by a PID closed loop algorithm. However no further information was provided as to what parameters define the spline itself. Therefore measurements from the real system were needed to be able to compare the performance of the optimal controller with the real system upon the same commanded task. Figure 2 shows the torque response of the real manipulator moving at maximum controller speed with no load. Superimposed noise effects were filtered out by means of a digital lowpass Butterworth filter.

For simulation purposes the system of non-linear equations describing the behaviour of the manipulator under the optimal control action was integrated numerically using a fourth order Runge-Kutta numerical algorithm. Figure 3 shows the simulated torque profile response. It can be seen that stretching the manipulator capabilities near maximum values dramatically improved the time taken to do the motion, demonstrating the performance of the optimal trajectory planner-controller. The algorithm has been applied to different points throughout the manipulator workspace with improvements in the range of 25-35%.

The value of the feedforward torque at the end of the motion corresponds to the actuator holding torque. It can be seen from comparison of graphs in figures 2 and 3 that in both cases the holding torque is very similar.

Concluding Remarks

An algorithm has been presented in which the role of manipulator dynamics in trajectory planning and control is investigated. Incorporating dynamics into the trajectory planning stage and applying optimal control theory has resulted in an al-

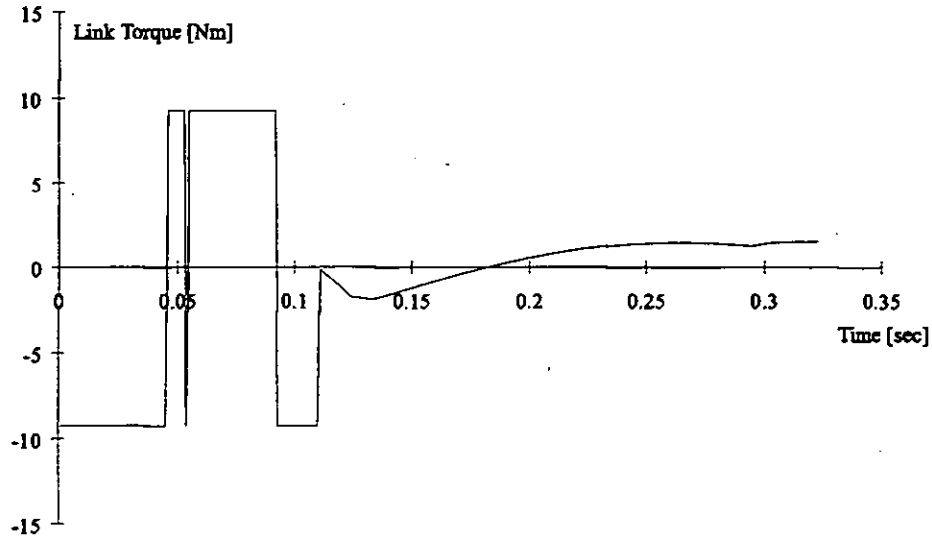

Figure 3: Simulated optimal torque response of robot joint.

gorithm that maximises the capabilities of the device, thus improving the overall manipulator time response by 25-35%. A feedforward control approach at the end of the motion has also been implemented to solve the problem of chattering around the end position. A comparison of the simulated optimal trajectory and actual measurements from an industrial manipulator, the CRS A251, has confirmed the validity of the strategy presented here.

In view of this fact a custom controller interface to the robot is currently being developed to test the practicality of a real-time implementation.

Acknowledgements

Jaime Valls Miro is currently undertaking a PhD in the area of robot motion control and simulation at Middlesex University and his research program is being sponsored by British Nuclear Fuels Plc.

References

An C.H. & Atkeson C.G. & Hollerbach J.M. 1988 Model-Based Ctrl. of a Robot Manip., MIT Press.

Kirk D.E. 1970 Optimal Ctrl Theory: an introduction, Prentice-Hall Inc.

Chen Y. & Chien S.Y.P 1992 General struct. of time-optimal ctrl. of robotic manip. moving along prescribed paths, *Proc. of the American Ctrl. Conf.* , **2**, 1510-1514.

Kim B.K. & Shin K.G. 1985 Suboptimal ctrl. of indust. manip. with a weighted minimum time-fuel criterion, *EEE Trans. on Autom. Ctrl*, **AC-30(1)**, 1-10.

Wiens G.J. & Berggren M.J. 1991 Suboptimal path planning of robots: minimal nonlinear forces and energy, *J. of Dyn. Sys., Meas., and Ctrl. Trans. of the ASME*, **113**, 748-752.

ON-LINE TIME-OPTIMAL ALGORITHM FOR MANIPULATOR TRAJECTORY PLANNING

J.V. Miró *, A.S. White, R. Gill

Middlesex University, Bounds Green Road, London N11 2NQ, U.K.
**Fax : +44(0)181 361 1726 and e-mail : j.v.miro@mdx.ac.uk*

Keywords : Robotics, Optimal Control, Trajectory Planning, Nonlinear Dynamics.

Abstract

A near-optimal solution to the path-unconstrained time-optimal trajectory planning problem is addressed in this paper. While traditional trajectory planning strategies are entirely based on kinematic considerations, manipulator dynamics are neglected altogether. This often results in mathematically tractable solutions which do not maximise the performance that manipulators might be capable of at any given time. The strategy presented in this work has two distinguishing features. First, the trajectory planning problem is reformulated as an optimal control problem which is in turn solved using Pontryagin's Maximum/Minimum Principle. This approach merges the traditional division of trajectory planning followed by trajectory tracking into one process. Secondly, the feedback form compensates for the dynamic approximation errors derived from the linearization approach taken and also the fundamental parameter uncertainty of the dynamic equations of motion. Results from simulations and an on-line implementation to a general purpose open-chain industrial manipulator, the CRS A251, confirm the validity of the approach and show that maximising the capabilities of the device can lead to an overall improvement in the manipulator time response of up to 25-30% .

1 Introduction

The basic problem in robotics is planning motions to solve some specific task and then controlling the response of the robot to achieve those motions. Depending on the literature it is common practice to refer to the `path` as the curve in space that the manipulator end-effector must visit during the motion with central attention to collision avoidance. A `trajectory` is defined as the time sequence of intermediate configurations of the arm along the programmed path. These configurations, together with their first and possibly second time derivatives are then fed to the servo mechanisms `controlling` the actuators that actually move the arm. Most current industrial manipulators follow these steps to accomplish a specified task because the overall motion process could become fairly complicated if considered in its entirety [1].

Traditional trajectory planning strategies such as low degree polynomials, cubic splines, linear functions with parabolic blends or bang-bang [2] are entirely based on kinematic considerations to satisfy a set of constraints on the position, velocity and acceleration of the arm at a number of points along the desired trajectory. Regarding control strategies, practically all industrial manipulators currently in use are based on classical linear feedback controllers with PD or PID algorithms.

However, it is well known that mechanical manipulators are multibody systems whose dynamic behaviour is described by strongly non-linear differential equations. Nonlinearities are associated both with position and velocity variables, and also their payloads. While a few more advanced model-based controllers [3] compensate for some of the position-dependent non-linear terms such as gravity they very often neglect the velocity-dependent terms in the controller design. Although industrial trackers can generally keep the manipulator fairly close to the desired trajectory [4], the simplistic division of robot motion into trajectory planning and tracking often results in mathematically tractable solutions which do not maximise the manipulator's maximum capabilities. The source of such underutilisation lies in the fact that both maximum speeds and accelerations/decelerations are limited for a given robot structure by the torque capacity of the joint actuators which vary across the workspace. Yet when trajectories are planned constant maximum bounds along each degree of freedom (DoF) are assumed. Thus, in order not to exceed the actual capabilities of the device, these specifications must be chosen conservatively, possibly forcing the robot to be underutilised [5].

A procedure to overcome these drawbacks is proposed in this paper by formulating the design of a trajectory planner as an optimal control problem in configuration space with account for the dynamics of the manipulator. This method removes the traditional inefficient assumption of trajectory planning and control as two separate motion stages. The work presented focuses on on-line minimum-

time point-to-point unconstrained motions, i.e., robotic applications which do not require manipulators to strictly follow a prescribed path or trajectory between points. Instead, collision avoidance is taken care of at task level by some means of specifying appropriate control points. Hence, the manipulator control problem can be addressed as a more general form in which the robot is given relative freedom to move along any trajectory between any two given intermediate or end path points.

Although the approach taken to solve the problem is similar to that adopted by some researchers in the past [4, 6, 7], in their work a geometric path is first assumed given as a function of a single parameter λ, $\theta = f(\lambda)$. Then the path is time parameterised as $\lambda(t)$ to minimise a pre-specified performance index (e.g., a function of time) subject to the dynamic constraints of the system. The computational complexity of these algorithms is such that they have been regarded for off-line use in the literature.

A more general approach to the truly unconstrained problem where the shape of the path is also optimised in the process was presented in [2], in which the use of geometric uniform cubic B-splines was extended to solve the optimal problem. The resulting non-linear programming problem was then solved with the sequential quadratic programming (SQP) numerical algorithm. The optimality of the solution however, depends on several factors such as an initial guess for the unknown spline coefficients and the traveling time. Furthermore, the iterative procedure renders it suitable for off-line use only.

The approach described here computes a trajectory that is optimal with respect to a timing performance index without going through an intermediate geometric path, and is also simple enough to allow for on-line implementation with actual personal computers. The work is similar to the optimal weighted feedback controller initially described and simulated in [8]. However, new alternatives towards an efficient on-line implementation of a time-optimal algorithm have been explored. In particular, a close look at the manipulator electro-mechanical parts and a more general form of dynamics linearisation provides a more realistic approach to the design of a truly on-line optimal unconstrained trajectory generator.

The validity of the algorithm presented in this paper has been established by numerical/graphical simulation of the first three joints of an industrial manipulator with five rigid links, the CRS A251. The accuracy of the simulations is demonstrated in a comparison to measurements performed on an implementation of the strategy on the real robot.

2 Problem Statement

The problem can then be summarized as follows: Using a closed form of manipulator dynamics [1] as shown in equation (1) find the optimal path that the robot tool centre point (TCP) should follow to move (in joint space) from the initial position q_{t_i} to the final position q_{t_f} while minimizing a performance index $J(t)$.

$$\boldsymbol{D(q)\ddot{q}+H(q,\dot{q})+G(q)=\tau} \tag{1}$$

where $\boldsymbol{D(q)}$ denotes the n x n link and actuator inertia matrix (n = number of DoF of the manipulator), $\boldsymbol{H(q,\dot{q})}$ is a n x 1 vector containing all velocity dependent terms arising from centripetal and coriolis forces and also effective viscous friction and $\boldsymbol{G(q)}$ represents the n x 1 gravity force terms. Joint torques are included in $\boldsymbol{\tau}$.

Depending on which parameters are considered in the cost function $J(t)$ the problem can be seen as an energy optimization problem, minimum time problem, a combination of both, acceleration optimization problem, etc... Efforts are being concentrated here into reducing the time required by the manipulator to achieve the desired location, i.e., the optimal time problem. Thus, the performance index can be expressed as:

$$J(t) = \int_{t_i}^{t_f} 1 dt \tag{2}$$

2.1 Non-linear Optimal Trajectory Planner

If the terms of equation (1) were expanded it would be easy to see that they are inherently nonlinear, both in position and velocity variables. This makes it impossible to design a trajectory planner-controller based on linear control theory. However, since equation (1) represents the nonlinear plant dynamics correctly, we can use this model to actively compensate for all nonlinearities as others researcher have done in the past [9]. To this end the problem was initially attempted by using modern optimal control theory [10]. Hence, a $2n$-dimensional state vector $\boldsymbol{y(t) = [y_1(t), y_2(t)]^T = [q, \dot{q}]^T}$ will be used to rewrite the dynamic equation (1) in state-space form as

$$\boldsymbol{\dot{y} = A(y) + B(y)\tau} \tag{3}$$

where

$$\boldsymbol{A(y)} = \begin{bmatrix} \boldsymbol{y_2} \\ \boldsymbol{-D^{-1}(y_1)[H(y_1,y_2) + G(y_1)]} \end{bmatrix}$$
$$\boldsymbol{B(y)} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{D^{-1}(y_1)} \end{bmatrix} \tag{4}$$

Each joint of an industrial manipulator is driven by separate actuators and therefore the optimal trajectory is designed under constraints imposed on the maximum torque bounds that can be exerted by the actuators, i.e., $\tau \leq |\tau^{\pm}|$. Given the initial and final states y_{t_i} and y_{t_f} this results in a non-linear two-point boundary-value problem whose solution is very difficult, if not impossible to obtain [8]. In order to overcome such difficulty an alternative approach is proposed where system equations are linearize and decoupled in terms of the joint variable q.

[1] derived from a Lagrangian formulation of mechanics

3 Linear Optimal Trajectory Planner

The dynamic model (3) can be first rearranged into a closed-form where there is a clear structure of the dependency between the acceleration of joint i and the actuator torque driving the joint as follows

$$\dot{y_{2_i}} = \alpha_i(\boldsymbol{y_1})\tau_i + \beta_i(\boldsymbol{y_1}, \boldsymbol{y_2}, \tau) \qquad i = 1 \ldots n \qquad (5)$$

where

$$\begin{aligned} \alpha_i(\boldsymbol{y_1}) &= D_{ii}^{-1}(\boldsymbol{y_1}) \\ \beta_i(\boldsymbol{y_1}, \boldsymbol{y_2}, \tau) &= \textstyle\sum_{j=1, j\neq i}^{n} D_{ij}^{-1}(\boldsymbol{y_1})\tau_j - \\ & \textstyle\sum_{j=1}^{n} D_{ij}^{-1}(\boldsymbol{y_1})[H_j(\boldsymbol{y_1}, \boldsymbol{y_2}) + G_j(\boldsymbol{y_1})] \end{aligned} \qquad (6)$$

D_{ij}^{-1} denotes the (i,j)th element of the inverse of the inertia matrix. Coefficients $\alpha_i(\boldsymbol{y_1})$ and $\beta_i(\boldsymbol{y_1}, \boldsymbol{y_2}, \tau)$ are time-variant non-linear functions of the manipulator position, velocity and input. However, equation (5) could be regarded as an uncoupled linear system by assuming α_i and β_i constant during each sample interval ΔT at which the trajectory is generated. It will be assumed [10] that the sampling interval is small enough so that the continuous control signal can be approximated by a piecewise-constant function that changes only at instants $t = 0, \Delta T, ..., (N-1)\Delta T$. That will allow to approximate the inertial coupling in the joint at time t_c with the last control input at time $t_c - \Delta T$ with a minimal overall error. Non-linear terms in $\boldsymbol{y_1}$ and $\boldsymbol{y_2}$ are considered in feedhack form by using the manipulator present state at the beginning of the current sampling period t_c. The resulting linear and decoupled equation for each link can be then described as

$$\dot{y_{2_i}} = \alpha_i\tau_i + \beta_i \qquad \text{for } i = 1 \ldots n \qquad (7)$$

which reduces the problem to the solution of a quasi-double integrator problem that must be solved in real-time at each sample interval. It is interesting to note here that in doing so the problem is most suitable to be treated with concurrent programming and that is an objective for the near future.

3.1 The Double Integrator Problem

The double integrator problem [10] is applied when the dynamic behaviour of the system can be described by the following set of ordinary differential equations:

$$\begin{aligned} dy_1(t)/dt &= y_2(t) \\ dy_2(t)/dt &= \tau(t) \end{aligned} \qquad (8)$$

where $y(t) = (y_1(t), y_2(t))$ represent the state vector for the unitary-mass system. It can be shown by applying Pontryagin's maximum principle [10] that a necessary condition to transfer the system from a specified initial point

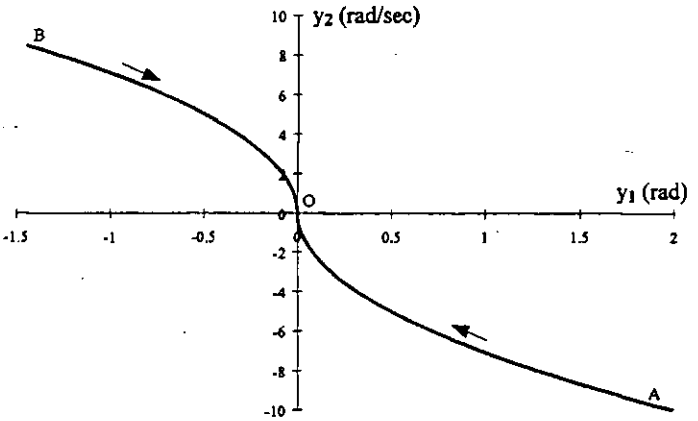

Figure 1: Double Integrator State Switching Curve.

$y(t_i)$ to a specified end point $y(t_f)$ in minimum time is to let the control variable $\tau(t)$ take one or other of its extreme values $\tau^{\pm}(t)$. It can also be shown that $\tau(t)$ changes sign not more than once during the whole motion, providing the optimal state switching curve illustrated in figure 1 where arrows show the direction of increasing time t. The switching curve AOB can be described as a function of the current state $y(t)$ by

$$y_1(t) = -\frac{y_2(t)|y_2(t)|}{2\tau^{+}(t)} \qquad (9)$$

An optimal trajectory consists then on two consecutive segments. In the first the state moves towards the switching curve AOB under the action of one of the two maximum control bounds. That is then followed by the opposite maximum control action which effectively slides the state along the switching curve towards the state origin. Hence, the time-optimal control law at any time t can be easily deduced in accordance with the following logical rules:

$$\tau(t) = \begin{cases} \tau^{+}(t) & \text{if } y(t) \text{ lies below } AOB \text{ or on } AO \\ \tau^{-}(t) & \text{if } y(t) \text{ lies above } AOB \text{ or on } BO \end{cases} \qquad (10)$$

It is interesting to note the symmetry of the optimal curve AOB with respect to both axis given the time-invariability of the system described by (8).

3.2 Structure of the Optimal Trajectory Planner-Controller

The approach described above was extended to solve the uncoupled quasi-double integrator prohlem described hy equation (7). However, the inclusion of the state-dependent manipulator dynamics makes the system time-invariant no longer, proving then necessary to update the optimal switching curve at each sample time [2]. Unlike before the optimal switching curve at each time t can not be

[2] that is a great burden for real-time purposes if the robot is to be operated at high speeds as is desirable

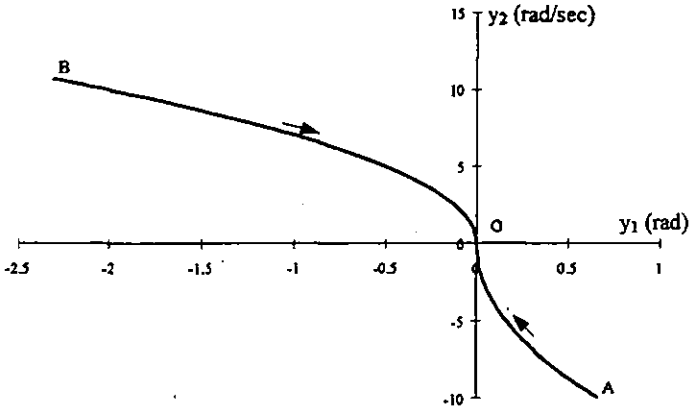

Figure 2: Quasi-Double Integrator State Switching Curve.

described by a unique equation, and therefore the control law for $i = 1 \ldots n$ needs to be considered for two separate cases depending on the current state:

$$\begin{cases} \text{if } y_{2_i}(t) \geq 0 \\ \quad \tau_i(t) = \begin{cases} \tau_i^+(t) & \text{if } y_{1_i}(t) < \dfrac{y_{2_i}^2(t)}{2\alpha_i(t)\tau_i^-(t) + \beta_i(t)} \\ \tau_i^-(t) & \text{otherwise} \end{cases} \\ \text{if } y_{2_i}(t) < 0 \\ \quad \tau_i(t) = \begin{cases} \tau_i^+(t) & \text{if } y_{1_i}(t) \leq \dfrac{y_{2_i}^2(t)}{2\alpha_i(t)\tau_i^+(t) + \beta_i(t)} \\ \tau_i^-(t) & \text{otherwise} \end{cases} \end{cases} \tag{11}$$

The behaviour of such control law at any instant time t can be depicted by the optimal state switching curve of figure 2 where the two cases are clearly identifiable by the non-symmetry of the optimal curve.

Since the dynamic model is updated with state feedback information at the beginning of each interval ΔT the approximation error described in Section 3 is implicitly compensated. Furthermore, at present time t_c the optimal control input should be determined based on the dynamic behaviour of the manipulator over the period $[t_c, t_f]$. Yet the dynamic coefficients are known for t_c and t_f but not for the period in between. It is therefore necessary to find a way to describe the overall dynamic behaviour of the system for the remaining of the actual motion on the basis of the current state and the final state. Some methods have been devised [8] which propose an arithmetic average of the dynamic coefficients. A more general form proposed in [9] has been adopted in which the overall dynamic behaviour of the manipulator is defined by two factors $\lambda_1, \lambda_2 \in [0, 1]$, one for each dynamic coefficient. This is shown in (12) for $i = 1 \ldots n$. The use of a factor allows for each boundary condition's dynamic performance to be weighted separately in the estimated final value. It also gives the flexibility to set weighting coefficients to different values (although the same values have been employed here for simplicity).

$$\begin{cases} \overline{\alpha_i(t_c)} &= (1-\lambda_1)\alpha_i(t_c) + \lambda_1\alpha_i(t_f) \\ \overline{\beta_i(t_c)} &= (1-\lambda_2)\beta_i(t_c) + \lambda_2\beta_i(t_f) \end{cases} \tag{12}$$

The resulting algorithm that yields the minimum-time trajectory and control action the robot should follow to move from the initial to the final state can be formulated as follows:

1. Derive the manipulator dynamic model. Lagrangian mechanics have been used here, but other alternatives are also viable. Most dynamic parameters have been estimated according to information from the manufacturer.

2. Given initial and final states compute the corresponding control actions from equation (1). The initial control action will act as the estimated control input into the algorithm.

3. Calculate dynamic behaviour for current state. That is described by $\alpha(q)$ and $\beta(q, \dot{q}, \tau)$ in equation (6). The linear approximation described by (7) is also applied now to obtain the linear behaviour of the system during the current time interval ΔT.

4. Average the dynamic coefficients according to (12) to achieve an overall dynamic performance of the manipulator. A constant λ=0.5 has been used here for both coefficients assigning the same weight to both states but further work to discern the significance of the weighting factors is currently being undertaken.

5. Use the current estimated dynamic model to update the switching curves according to the control law described by (11) and compute the optimal control action.

6. Go back to point 3 if final state has not been reached.

A well-known problem with any bang-bang method is control chattering in the vicinity of the target state caused by frequent switchings of the control input. Different alternatives to alleviate this undesirable effect have been proposed in the literature, e.g., the use of a smoothing function [11] or switching to a linear controller when the manipulator is within a prescribed range of the end state [8]. In the work presented here the possibility of applying feedforward torque control [3] around the desired end state has been sought with successful results. Therefore point 6 of the algorithm is further extended to compare current state with target state. If the difference is less than some state bounds specified by the user, the feedforward controller comes into action.

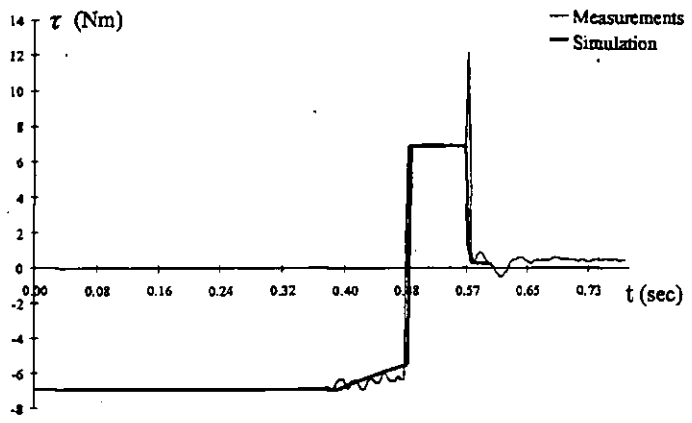

Figure 3: Torque joint 1

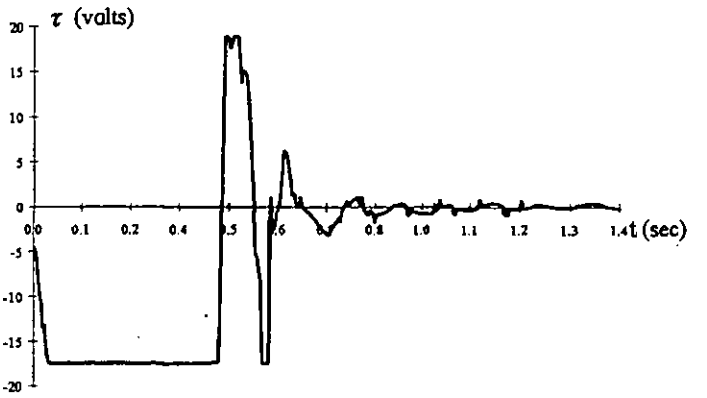

Figure 5: CRS PID joint 1

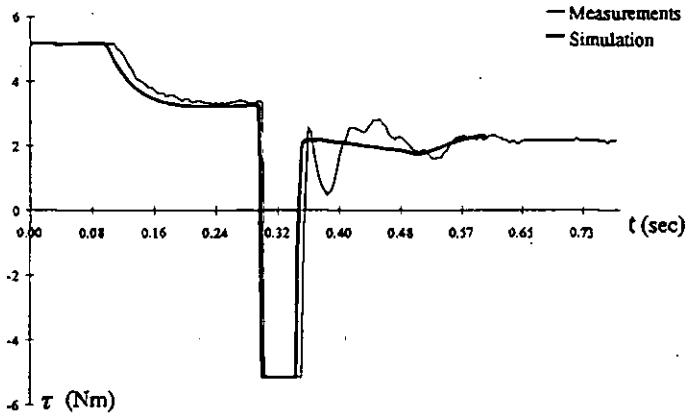

Figure 4: Torque joint 2

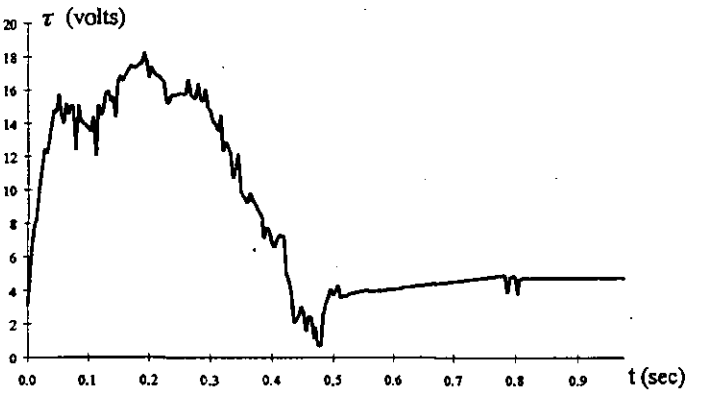

Figure 6: CRS PID joint 2

4 Simulation and Experimental Results

To simulate the structure of the proposed algorithm the system of non-linear equations that describes the behaviour of the manipulator under the optimal control action was integrated numerically using the fourth order Runge-Kutta algorithm. The program was written in ANSI C and linked to an advanced graphical robotic simulation environment running under UNIX [3]. The kinematics, dynamics and CAD models of the CRS A251 industrial manipulator were also developed and linked to the simulator. For more details on the actual implementation the reader may refer to [12].

The experimental equipment setup included a Pentium PC@75Mhz with one Lab-PC+ data acquisition card and a purpose-built PCB dedicated for each joint to increase processing speed. The PCB design holds mainly the digital counters required to obtain state feedback information from the industrial controller, along with analog outputs for the drive voltages to the motor power amplifiers.

Since the wrist joints are usually dominated by inertia, with gravity and inertial coupling effects in the range of one or two orders of magnitude down [13], only the gross motion links (waist, upper and fore arm) have been considered here for optimisation. Furthermore, due to lack of space results from joint 1 (waist) will not be presented here. This is not detrimental to the overall understanding of the paper, moreover when the motion of joint 1 is in a gravitational-free plane.

As a basis for comparison, the same joint path end points were presented to the simulation and experimental environment, and also the commercial controller. The experiments were computed at a sampling rate of 250 Hz which proved to be fast enough for the computational burden of the algorithm. Figures 3 and 4 demonstrate the accuracy of the simulation by superimposing a thinner curve gained from measurements to the bold curve gained from simulation of the torques exerted by the optimal trajectory planner. Some general conclusions can be drawn. It can be seen how due to the inherent performance of the actuators, the nominal torque is also dependent on the state of the system. Hence, saturation of the actuators must also be taken into account to achieve an accurate simulation yet the feedback nature of the algorithm allows for

[3] UNIX is a trademark of Bell laboratories.

constant bounds to be assumed. Whether variable state-dependent torque bounds would make any improvement to the optimal algorithm is still being investigated. Also, although lower feedback gains can be applied due to the use of a feedforward strategy around the end state, the change to the feedforward controller is still clearly manifested in Figure 3 with a high torque peak nearby the end position. This, however, is only kept for a very short period of time and does not contribute notably to the overall timing. In contrast, higher gains reduce the steady state error to around $+/-0.005$ link radians.

Figures 5 and 6 show the actuator response in Volts of the real manipulator moving from the same initial to end points at maximum actuator speed with no load. The commercial controller follows the traditional motion pipeline, i.e., an initial spline joint interpolated trajectory between path end points with command updates at 18 millisecond intervals followed by a PID closed loop algorithm at a servo rate of 1kHz. Despite an expected null steady-state error in the commercial manipulator, the optimal algorithm proves superior regarding timing constraints as a comparison of Figures 3, 4 and 5, 6 demonstrate. This makes the algorithm most suitable for gross motion when speed becomes the parameter to watch.

5 Concluding Remarks

An algorithm has been presented in which the role of manipulator dynamics in trajectory planning and control is investigated. Incorporating dynamics into the trajectory planning stage and applying optimal control theory has resulted in a maximisation of the device capabilities, thus improving the time response by up to 25-30%.

The scheme has been oriented towards an on-line implementation in which its closed-loop feedback form and simplistic linearisation approach has proved clearly advantageous to cope with the complexity of the robot dynamic model. A feedforward controller has also been proposed to remove the undesirable chattering in the vicinity of the end state.

A comparison of the simulated results with actual measurements from an experimental setup has confirmed the validity of the strategy presented here.

Acknowledgements

Jaime V. Miró is currently undertaking a PhD in the area of robot motion and simulation at Middlesex University and his research program is partially sponsored by British Nuclear Fuels Plc.

References

[1] Brady M., Hollerbach J.M., Johnson T.L., Lozano-Perez T., and mason M.T. *Robot Motion: planning and control.* MIT Press, (1982).

[2] Chen Y.C. Solving robot trajectory planning problems with uniform cubic b-splines. *Optimal Control Applications & Methods*, 2:247–262, (1991).

[3] An C.H., Atkeson C.G., and Hollerbach J.M. *Model-Based Control of a Robot Manipulator.* MIT Press, (1988).

[4] Shin K.G. and McKay N.D. Minimum-time control of robotic manipulators with geometric path constraints. *IEEE Transactions on Automatic Control*, **AC-30**(6):531–541, June (1985).

[5] Kim B.K. and Shin K.G. Minimum-time path planning for robot arms and their dynamics. *IEEE Transactions on Systems, Man and Cybernetics*, **SMC-15**(2):213–223, March/April (1985).

[6] Bobrow J.E., Dubowsky S., and Gibson J.S. On the optimal control of robotic manipulators with actuator constraints. In *Proceedings of the American Control Conference*, pages 782–787, June (1983).

[7] Pfeiffer F. and Johanni R. A concept for manipulator trajectory planning. *IEEE Journal of Robotics and Automation*, **RA-3**(2):115–123, April (1987).

[8] Kim B.K. and Shin K.G. Suboptimal control of industrial manipulators with a weighted minimum time-fuel criterion. *IEEE Transactions on Automatic Control*, **AC-30**(1):1–10, January (1985).

[9] Wiens G.J. and Berggren M.J. Suboptimal path planning of robots: minimal nonlinear forces and energy. *Journal of Dynamic Systems, Measurement, and Control. Transactions of the ASME*, **113**:748–752, December (1991).

[10] Kirk D.E. *Optimal Control Theory: an introduction.* Prentice-Hall Inc., (1970).

[11] Krejnin G.V., Sattar T.P., and Smelov L.A. Towards increasing the speed of manipulation mechanisms. *Journal of Machinery Manufacture and Reliability*, 3:62–68, (1994).

[12] Miro J.V., Stoker M., and Gill R. The use of computer graphics for robot motion simulation. In *The 11th ISPE/IEE/IFAC International Conference on CAD/CAM, Robotics & Factories of the Future*, pages 884–889, August (1995).

[13] Corke P.I. An automated symbolic and numeric procedure for manipulator rigid-body dynamic significance analysis and simplification. In *IEEE international conference on robotics and autmation*, pages 1018–1023, April (1996).